This document presents exercises for the week. The intention is for you to complete a selection of the exercises at your convenience and at your own pace.
Note: there is no expectation that we will tackle all of the exercises during the teaching week.

Propositional logic

Exercise 1

What are the truth values of the following (assuming (a) that x is a natural number, i.e. $x \in \mathbb{N}$, and (b) that pigs can't fly)?

(a) $(x = x) \Rightarrow$ (pigs can fly)

(b) $(x \neq x) \Rightarrow$ (pigs can fly)

(c) (pigs can fly) $\Rightarrow (x = x)$

(d) (pigs can fly) $\Rightarrow (x \neq x)$

Exercise 2

Construct a truth table for each of the following.

(a) $(p \wedge q) \Rightarrow p$

(b) $(\neg p \Rightarrow (p \wedge q)) \Leftrightarrow p$

(c) $(p \wedge (p \Rightarrow q)) \Rightarrow q$

Exercise 3

Present proofs of each of the following.

(a) $(p \Rightarrow \neg p) \Leftrightarrow \neg p$

(b) $(\neg p \Rightarrow p) \Leftrightarrow p$

(c) $(p \Rightarrow (q \Rightarrow r)) \Leftrightarrow ((p \wedge q) \Rightarrow r)$

(d) $(q \Rightarrow (p \Rightarrow r)) \Leftrightarrow (p \Rightarrow (q \Rightarrow r))$

(e) $((p \wedge q) \Leftrightarrow p) \Leftrightarrow (p \Rightarrow q)$

(f) $((p \vee q) \Leftrightarrow p) \Leftrightarrow (q \Rightarrow p)$

Exercise 4

Which of the following statements are tautologies?

(a) $(p \vee q) \Leftrightarrow ((\neg p \vee \neg q) \wedge q)$

(b) $(p \vee q) \Leftrightarrow ((\neg p \wedge \neg q) \vee q)$

Predicate logic

Exercise 5

Capture the following predicates about dogs. You should feel free to assume the existence of a type *Dog* and predicates such as *gentle*.

(a) "Some dogs are gentle and have been well trained."

(b) "All dogs are attractive if they are neat and well trained."

(c) "Some dogs are gentle only if they have been groomed by every trainer."

Exercise 6

The predicate p with free variables x and y is defined by

$$p \Leftrightarrow x + y > x^2$$

Which of the following are true?

(a) $\forall\, x : \mathbb{N} \bullet \exists\, y : \mathbb{N} \bullet p$

(b) $\exists\, y : \mathbb{N} \bullet \forall\, x : \mathbb{N} \bullet p$

(c) $(\forall\, x : \mathbb{N} \bullet \exists\, y : \mathbb{N} \bullet p) \Rightarrow (\exists\, y : \mathbb{N} \bullet \forall\, x : \mathbb{N} \bullet p)$

(d) $(\exists\, y : \mathbb{N} \bullet \forall\, x : \mathbb{N} \bullet p) \Rightarrow (\forall\, x : \mathbb{N} \bullet \exists\, y : \mathbb{N} \bullet p)$

Exercise 7

Define a pair of predicates, p and q, involving a free variable, x, such that

(a) $\forall\, x : \mathbb{N} \bullet p$ is equivalent to *false* but $\exists\, x : \mathbb{N} \bullet p$ is equivalent to *true*, and

(b) $\exists\, x : \mathbb{N} \bullet p \Rightarrow q$ is equivalent to *true* but $\forall\, x : \mathbb{N} \bullet p \Rightarrow q$ is equivalent to *false*.

As an example, if p is the predicate $x = x$, then $\forall\, x : \mathbb{N} \bullet p$ and $\exists\, x : \mathbb{N} \bullet p$ are both equivalent to *true*.

Exercise 8

Determine the results of each of the following, using renaming to avoid variable capture as appropriate.

(a) $(\forall\, x : \mathbb{N} \bullet x \geq y)\,[z/y]$

(b) $(\forall\, x : \mathbb{N} \bullet x \geq y)\,[x + y/y]$

(c) $(y > 0 \land \forall\, x : \mathbb{N} \bullet x \geq y)\,[x + 3/y]$

Equality

Exercise 9

Use the one-point rule for existential quantification to simplify each of the following statements.

(a) $\exists\, y : \mathbb{N} \bullet y \in \{0, 1\} \land y \neq 1 \land x \neq y$

(b) $\exists\, x, y : \mathbb{N} \bullet x + y = 4 \land x < y$

(c) $\forall\, x : \mathbb{N} \bullet \exists\, y : \mathbb{N} \bullet x = y + 1$

(d) $\exists\, x : \mathbb{N} \bullet (x = 1 \land x > y) \lor (x = 2 \land x > z)$

Exercise 10

Can you provide an example to describe the relationship that exists between $\exists_1$ and μ?

Exercise 11

Which of the following statements are provable in our mathematical language?

(a) $(\mu\, a : \mathbb{N} \mid a = a + a) \;=\; 0$

(b) $(\mu\, b : \mathbb{N} \mid b = b * b) \;=\; 1$

(c) $(\mu\, c : \mathbb{N} \mid c > c + c) \;=\; (\mu\, c : \mathbb{N} \mid c > c + c)$

(d) $(\mu\, d : \mathbb{N} \mid d = d \div d) \;=\; 1$

Exercise 12

Suggest μ-expressions for each of the following.

(a) The height of the highest mountain.

(b) The number of words in the second longest chapter.

(c) The difference between n and 100, where n is the largest multiple of 8 that is less than 100.

You should feel free to assume the existence of any relevant sets and functions (the function 'second longest chapter' not being one of them!).

Deductive proofs

Exercise 13

By exhibiting a proof tree, show that the following is a theorem of our natural deduction system.

$$(p \wedge (p \Rightarrow q)) \Rightarrow (p \wedge q)$$

Exercise 14

By exhibiting a proof tree, show that the following is a theorem of our natural deduction system.

$$((p \wedge q) \Leftrightarrow p) \Leftrightarrow (p \Rightarrow q)$$

Exercise 15

By exhibiting a proof tree, show that the following is a theorem of our natural deduction system.

$$((p \Rightarrow q) \wedge \neg q) \Rightarrow \neg p$$

Exercise 16

By exhibiting a proof tree, show that the following is a theorem of our natural deduction system.

$$(p \wedge (q \vee r)) \Leftrightarrow ((p \wedge q) \vee (p \wedge r))$$

Exercise 17

By exhibiting a proof tree, show that the following is a theorem of our natural deduction system.

$$(p \vee (q \wedge r)) \Leftrightarrow ((p \vee q) \wedge (p \vee r))$$

Exercise 18

By exhibiting a proof tree, show that the following is a theorem of our natural deduction system.

$$(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$$

Sets and types

Exercise 19

Which of the following statements are true?

(a) $1 \in \{4, 3, 2, 1\}$

(b) $\{1\} \in \{1, 2, 3, 4\}$

(c) $\{1\} \in \{\{1\}, \{2\}, \{3\}, \{4\}\}$

(d) $\emptyset \in \{1, 2, 3, 4\}$

Exercise 20

List *all* of the elements of the following sets.

(a) $\{1\} \times \{2, 3\}$

(b) $\emptyset \times \{2, 3\}$

(c) $(\mathbb{P}\,\emptyset) \times \{1\}$

(d) $\{(1, 2)\} \times \{3, 4\}$

Exercise 21

Assume that $*$ denotes multiplication, *div* denotes integer division and *mod* denotes remainder. Describe each of the following using set comprehension.

(a) The set of integers between 0 and 100 inclusive.

(b) The set of integer multiples of 10.

(c) The set of integers divisible by 2, 3, or 5.

Exercise 22

We might define the set $\{0, 1, 2, 3, 4\}$ via $\{n : \mathbb{N} \mid n \leq 4\}$. Represent each of the following sets in terms of set comprehension.

(a) $\{0, 1, 4, 9, 16\}$

(b) $\{(0, 0), (1, 1), (2, 4), (3, 9), (4, 16)\}$

(c) $\{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$

(d) $\{(\emptyset, 0), (\{0\}, 1), (\{1\}, 1), (\{0, 1\}, 2)\}$

Exercise 23

To show that two sets are equal, we must establish that they have exactly the same elements. A suitable proof will involve establishing an equivalence of the form

$$x \in a \Leftrightarrow x \in b$$

generalising the result, and applying the rule 'extension'.

For example, we may show that

$$\begin{aligned} & x \in (a \cup b) \\ & \quad \Leftrightarrow x \in a \vee x \in b && \text{[union]} \\ & \quad \Leftrightarrow x \in b \vee x \in a && \text{[commutativity of } \vee\text{]} \\ & \quad \Leftrightarrow x \in (b \cup a) && \text{[union]} \end{aligned}$$

and hence that $a \cup b = b \cup a$. Using this approach, show that the following are all true.

(a) $a \cap a = a$

(b) $a \cup a = a$

(c) $a \cap \emptyset = \emptyset$

(d) $a \cup \emptyset = a$

(e) $a \cap (b \setminus a) = \emptyset$

(f) $a \cup (b \setminus a) = a \cup b$

(g) $a \setminus (b \cup c) = (a \setminus b) \cap (a \setminus c)$

(h) $a \setminus (b \cap c) = (a \setminus b) \cup (a \setminus c)$

Definitions

Exercise 24

Using basic type declarations and abbreviations, introduce expressions to represent each of the following. You should, of course, feel free to make use of $\mathbb{Z}$.

(a) The set of all pairs of integers.

(b) The set of all pairs of integers in which each element is strictly greater than zero.

(c) The set of all people.

(d) The set of all couples, where a couple comprises exactly two people.

(e) The set of all parties, where a party is any group of eight or more people.

Exercise 25

In our use of the empty set symbol, we will invariably omit the actual parameter. By adding suitable parameters constructed from $\mathbb{N}$, show how it is possible for the following expressions to be well-defined.

(a) $\emptyset \in \emptyset$

(b) $\emptyset \subseteq (\emptyset \times \emptyset)$

(c) $(\emptyset \times \{\emptyset\}) \subseteq \emptyset$

Exercise 26

The symbol $\notin$ is not a primitive object: it can be defined in terms of the symbol for set membership $\in$. Using a generic definition, define $\_ \notin \_$ to be the set of pairs (x, s) such that x is not an element of s.

Relations

Exercise 27

Write the following relations as sets in extension (that is, as a list of elements between braces).

(a) $\{0, 1\} \leftrightarrow \{0, 1\}$

(b) $\{0\} \leftrightarrow \{0, 1\}$

(c) $\emptyset \leftrightarrow \{0, 1\}$

(d) $\emptyset \leftrightarrow (\emptyset \leftrightarrow \emptyset)$

Exercise 28

Assuming that R is the relation

$$\{0 \mapsto 1, 0 \mapsto 2, 0 \mapsto 3, 1 \mapsto 2, 1 \mapsto 3, 2 \mapsto 3\}$$

what do the following sets look like in extension?

(a) $\operatorname{dom} R$

(b) $\operatorname{ran} R$

(c) $\{1, 2\} \lhd R$

(d) $R \rhd \{1, 2\}$

Exercise 29

The relations R and S are defined as follows.

$$R == \{1 \mapsto 4, 2 \mapsto 3, 3 \mapsto 2, 3 \mapsto 3, 4 \mapsto 1\}$$
$$S == \{1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 4\}$$

Write each of the following sets in extension.

(a) $R \fatsemi S$

(b) $S \fatsemi R$

(c) $R \fatsemi R$

(d) $R \fatsemi R \fatsemi R$

Exercise 30

If a and b are elements of the set $Person$, then we write $a\ childOf\ b$ to indicate that a is a child of b. Using the relation $childOf$ and our relational operators, define the following.

(a) The relation $parentOf$, which holds between two people exactly when the first is the parent of the second.

(b) The relation $siblingOf$, which holds between two people exactly when they have the same parent.

(c) The relation $cousinOf$, which holds between two people exactly when a parent of one is a sibling of the parent of the other.

(d) The relation $ancestorOf$, which holds between two people exactly when the first is an ancestor of the second.

Exercise 31

If R is defined by

$$R == \{ n : \mathbb{N} \bullet n \mapsto n + 1 \}$$

identify the

(a) reflexive,

(b) symmetric,

(c) transitive and

(d) reflexive transitive closures

of R.

Exercise 32

If R, S, A, B and C are defined by

$$\begin{array}{l} R, S : X \leftrightarrow Y \\ A, B : \mathbb{P}\, X \\ C : \mathbb{P}\, Y \end{array}$$

show that each of the following is true.

(a) $A \lhd (B \lhd R) = (A \cap B) \lhd R$

(b) $(R \cup S) \rhd C = (R \rhd C) \cup (S \rhd C)$

(c) $(A \setminus B) \lhd R = (A \lhd R) \setminus (B \lhd R)$

Functions

Exercise 33

We may categorise the nine elements of $\{0, 1\} \nrightarrow \{0, 1\}$ according to whether they are injective, surjective, both, or neither. State which of these nine functions are

(a) total,

(b) neither injective nor surjective,

(c) injective but not surjective,

(d) surjective but not injective, or

(e) bijective.

Exercise 34

Consider the following sets.

$$A = \{(1, a), (2, b), (3, c), (4, d)\}$$
$$B = \{(2, b), (4, b)\}$$
$$C = \{x : A \mid (x.1 = 1 \lor x.1 = 3) \bullet (x.1, c)\}$$

Calculate the following.

(a) $A \oplus B$.

(b) $A \oplus C$.

(c) $A \oplus (C \oplus B)$.

(d) $(A \oplus C) \oplus B$.

Exercise 35

Consider again the relation *parentOf*, which holds between two people exactly when the first is the parent of the second. Define, via axiomatic definitions, the following functions in terms of *parentOf*.

(a) $children : Person \to \mathbb{P}\ Person$, such that, for every person, p, $children\,(p)$ denotes the set containing p's children.

(b) $number\_of\_grandchildren : Person \to \mathbb{N}$, such that, for every person, p, $number\_of\_grandchildren\,(p)$ denotes the number of grandchildren that p has.

Exercise 36

Recall the *drives* relation from the Relations lecture.

Define a function, *number_of_drivers*, that takes as input an element, r, of type $Drivers \leftrightarrow Cars$, and returns as output function that maps each car that appears in r to the number of people in r who drive that car.

So, for example,

$$number\_of\_drivers\,(drives) = \{alfa \mapsto 1, beetle \mapsto 2, cortina \mapsto 1\}$$

Sequences

Exercise 37

Consider the following sequences.

$$r = \langle a, b, c, d \rangle$$
$$s = \langle a, a, b \rangle$$
$$t = \langle a, b, c, d \rangle$$

Given the above sequences, calculate the following.

(a) $r \cap s$

(b) $r \cup s$

(c) $r \setminus s$

(d) dom t

(e) ran s

(f) $r^{\sim}$

(g) $\{1, 2\} \lhd r$

(h) $s \rhd \{a\}$

(i) $(t \oplus s)(\!| \{1, 2\} |\!)$

(j) $r\,3$

Exercise 38

Recall the *trains* example from the Functions and Sequences lecture. There, we had a sequence, $trains \in \text{seq}\,(Place \times Place)$, with each pair of the form (p_1, p_2) indicating that a train starts at p_1 and ends at p_2. Define the following.

(a) A function that take an element $p \in Place$ as input and returns a set containing the places to which p maps in *trains*.

(b) A set containing elements of *Place* that appear exactly once as a destination in *trains*.

(c) The element of *Place* that appears the most times as a destination in *trains*. (You should feel free to assume that there is a unique such element.)

Exercise 39

Consider the following.

$$Coin ::= large \mid medium \mid small$$

$$Collection == \text{bag}\ Coin$$

(a) Define a function, $large\_coins \in Collection \to \mathbb{N}$, which, given some $c \in Collection$, returns the number of instances of *large* in c. So, for example

$$large\_coins\,(\llbracket\, large, large, medium, small \,\rrbracket) = 2$$

(b) Define a function, $add\_coin \in Collection \times Coin \to Collection$, which, given some $c \in Collection$ and some $d \in Coin$, returns a bag that is effectively c with d added. So, for example

$$add\_coin\,(\llbracket\, large, large, medium, small \,\rrbracket, small) = \\ \llbracket\, large, large, medium, small, small \,\rrbracket$$

Modelling

The following questions represent a *significant* 'step up' in terms of difficulty. They would normally be undertaken via groups, taking most of Thursday.

The exercises serve two purposes. First, they 'bring together' many of the concepts taught during the week. Second, they give a feel for the kinds of questions you might expect to see in the course assignment.

In conclusion: if you feel like it takes a long time to get started, don't worry, as (a) that's expected and (b) you will not be alone.

Exercise 40

A TV recording system records television programmes to a hard-drive. The hard-drive has the capacity to store up to 200 hours of programming; each programme may be at most 6 hours in length.

When the viewer accesses the hard-drive, they are presented with a menu presenting all of the shows currently stored. The details are:

- title;
- programme length; and
- whether or not the programme has been viewed.

You may assume the following types and abbreviations.

$[Title]$
$Length == \mathbb{N}$
$Viewed ::= yes \mid no$

(We shall assume that the length of recordings is represented in terms of **minutes**.)

(a) Complete the following axiomatic definition with appropriate constraint information ("the hard-drive has the capacity to store up to 200 hours of programming; each programme may be at most 6 hours in length"):

$$hd : \operatorname{seq}(Title \times Length \times Viewed)$$
$$\vdots$$

The sequence hd captures information pertaining to programmes stored on the hard-drive. (You should assume, for now, the existence of a function $cumulative\_total \in \operatorname{seq}(Title \times Length \times Viewed) \to Length$. This function will be defined in part (d).)

(b) Define, via set comprehension, the collection of titles of programmes (which appear in hd) that are over two hours in length.

(c) Define functions *viewed* and *not_viewed* that take sequences of type $Title \times Length \times Viewed$, and return the sequences with the not viewed and viewed programmes removed respectively. So,

$$viewed\ \langle (t_1, 3, yes), (t_2, 4, yes), (t_1, 5, no) \rangle = \langle (t_1, 3, yes), (t_2, 4, yes) \rangle$$
$$not\_viewed\ \langle (t_1, 3, yes), (t_2, 4, yes), (t_1, 5, no) \rangle = \langle (t_1, 5, no) \rangle$$

(d) Define a **recursive** function, *cumulative_total*, that takes sequences of type $Title \times Length \times Viewed$ and returns the cumulative length of the programmes recorded. So,

$$cumulative\_total\ \langle (t_1, 3, yes), (t_2, 4, yes), (t_1, 5, no) \rangle = 12$$

(e) Give a μ-expression for the title of the longest programme that appears in hd.

(f) Define a function that maps programme titles (which appear in hd) to cumulative lengths, i.e.,

$$f\ \langle (t_1, 3, yes), (t_2, 4, yes), (t_1, 5, no) \rangle = \{t_1 \mapsto 8, t_2 \mapsto 4\}$$

(g) Define a function that takes a sequence of programmes and removes the longest viewed one, i.e.,

$$g\,\langle (t_1, 3, yes), (t_2, 4, yes), (t_1, 5, no)\rangle = \langle (t_1, 3, yes), (t_1, 5, no)\rangle$$

(h) Define a function that takes an element of $\mathrm{seq}\,(\mathit{Title} \times \mathit{Length} \times \mathit{Viewed})$ and sorts that sequence in terms of programme length—with the longest programme appearing first. So,

$$s\,\langle (t_1, 3, yes), (t_2, 4, yes), (t_1, 5, no)\rangle = \langle (t_1, 5, no), (t_2, 4, yes), (t_1, 3, yes)\rangle$$

Exercise 41

Imagine that a table is presented at the end of each calendar year, summarising the details of courses. The table is of the following form:

Date	*Course*	*Location*	*Lecturer*	*Attendees*	*Submissions*	*Average*
16/*Jan*/23	*DES*	479	*IF*	10	9	60
23/*Jan*/23	*CDS*	479	*JD*	10	10	55
23/*Jan*/23	*NES*	478	*AP*	10	11	65

Here: the date represents the start date of the five day course; the course column uses a three letter acronym to identify the course; the location is one of two teaching centres (denoted by 478 or 479); the lecturer column identifies the lead lecturer; column 5 details the number of attendees; column 6 details the number of assignment submission (which may be greater than the number of attendees); column 7 gives the grade average for the course.

We assume, then, the existence of types *Date*, *Course*, *Location*, *Lecturer* and *Grade*, and introduce

$$\mathit{Entry} == \mathit{Date} \times \mathit{Course} \times \mathit{Location} \times \mathit{Lecturer} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$$
$$\mathit{Table} == \mathrm{seq}\ \mathit{Entry}$$

The axiomatic definition *records*, given by

$$\begin{array}{|l} \mathit{records} : \mathit{Year} \pfun \mathit{Table} \\ \hline \vdots \end{array}$$

captures the notion of tables—sequences of entries—being associated with years.

(a) Show how this axiomatic definition may be completed via the enforcement of the following constraints.

(i) *records* contains entries for the years between 1993 and *current*—and no others. (Assume that *current* has been declared as a global variable of type *Year*.)

(ii) No more than 50 courses may run in any given year.

(iii) For every year, y, the entries in *records* y pertain to courses starting in that year. (Assume the existence of a function $year : Date \to Year$, which maps dates to years. So, for example, $year\,(14/Jan/08) = 2008$.)

(iv) Teaching rooms can't be 'double booked', i.e., it should be impossible for a teaching room to be associated with two different courses in the same week.

(b) Define set comprehensions that capture the following information.

(i) The details of all course instances taught in room 479.

(ii) The details of all course instances where the number of submissions is greater than the number of attendees.

(iii) The details of all course instances with average grades of 70 or above.

(iv) The acronyms of all courses for which the average grade of each instance of that course is 70 or above.

(v) A set of pairs of type $Year \times \mathbb{P}\ Lecturer$, such that (y, L) appears in the set if, and only if, L is the set of lecturers who taught more than 6 courses in year y.

(c) Define a **recursive** function, 479_*courses*, that takes a sequence of type *Entry* and returns the details of courses taught in 479.

(d) Define a **recursive** function, *total*, that takes sequences of type *Entry* and returns the total number of all course attendees. So, for the simple table shown above, the function would return the value 30.

Exercise 42

We wish to model a particular component of a supermarket: the collection of checkout points at which customers may queue in order to pay for goods and leave the store. At each checkout, we are interested only in the sequence of people queueing to be served.

For the purposes of this question, then, a supermarket is modelled as a sequence of checkouts, and a checkout is modelled as a sequence of people, elements of some given type *Person*. In our basic model, there may be any number of people in a queue, and there may be any number of checkouts.

However, no one person may be in more than one queue at the same time.

(a) Using an axiomatic definition, define the set *State*, comprising all the possible values of the sequence of sequences representing the checkouts: that is, the possible states of the checkout component of the supermarket.

(b) Define a function that takes as arguments a queue number, a person, and a supermarket state (one of the elements of *State*) and returns a new supermarket state. The application of the function should model the effect of a person joining the specified queue.

Exercise 43

An on-line booking system for a group of cinemas reserves seats for individuals. We can capture this information for each performance associated with the group of cinemas in terms of an axiomatic definition:

$$\begin{array}{|l} bookings : Cinema \times Film \times Date \times Time \pfun \\ \qquad\qquad \mathbb{P}\,(Customer \times \mathbb{N} \times \mathbb{N}) \\ \hline \vdots \end{array}$$

That is, every performance is mapped to a set of triples of the form (c, m, n), where $c \in Customer$, m is the lower bound of the tickets bought by c and n is the upper bound. If, for example, a person bought one ticket, then $m = n$.

(a) Write the following constraints on the set of possible values in terms of predicate logic. (You may assume the existence of a function $max : Cinema \pfun \mathbb{N}$ that maps a cinema to the number of seats available in that cinema. You may also assume that the range of seats for each cinema c runs from 1 to $max\ c$.)

(i) No seat for a given performance can be sold to more than one customer.
(ii) All seats sold are in the range $1\,..\,max\ c$ for the cinema c that the booking has been made for.
(iii) For each block booking, the upper bound of seats sold is at least as great as the lower bound.
(iv) No two reserved blocks of seats can overlap.

(b) Show how the following information can be derived from *bookings* in terms of set comprehensions.

(i) The set of films that have been shown at the Banbury cinema.
(ii) A set, that contains pairs, the first component of which is the name of a film and the second component of which is itself a pair, with, in position 1 the name of a cinema and in position 2 the total number of days that the film was shown at that cinema.

Free types and induction

Exercise 44

The *reverse* function is anti-distributive:

$$reverse\,(s \frown t) \;=\; reverse\,t \frown reverse\,s$$

Prove this using the induction principle for sequences and the definition below.

$$reverse\,\langle\rangle \;=\; \langle\rangle \qquad \text{(reverse.1)}$$

$$reverse(\langle x\rangle \frown s) \;=\; (reverse\,s) \frown \langle x\rangle \qquad \text{(reverse.2)}$$

Exercise 45

Prove, using the induction principle for sequences, that

$$reverse\,(reverse\,s) \;=\; s$$

for any sequence s.

Exercise 46

The set of all binary trees with natural number leaves is defined by

$$Tree \;::=\; \mathsf{stalk} \mid \mathsf{leaf}\langle\!\langle \mathbb{N} \rangle\!\rangle \mid \mathsf{branch}\langle\!\langle Tree \times Tree \rangle\!\rangle$$

(a) Define a function *count* that returns the total number of leaves in an element of *Tree*.

(b) Define a function *flatten* that maps a tree to a sequence of natural numbers while preserving the multiplicity of each number that appears.

Exercise 47

Show that, for any tree t:

$$\#(flatten\,t) \;=\; count\,t$$

Supplementary material: assignment practice

All of the exercises serve, in some sense, as 'practice for the assignment' — even though their primary purpose is to supplement learning. The following questions, however, are explicitly 'practice for the assignment' as they are drawn from a previous assignment.

Exercise 48

You have been asked to design (via a formal model) a prototype for a new online music platform called *Streamify*. (Note: all of the remaining questions will leverage the definitions of this question.)

Three basic types are of initial interest: $PlaylistId$, $SongId$ and $UserId$.

$$[PlaylistId, SongId, UserId]$$

A *playlist* consists of an injective, non-empty sequence of songs:

$$Playlist == \mathrm{iseq}_1\ SongId$$

The initial specification of the system is as follows.

$$\begin{array}{|l} songs : \mathbb{F}\ SongId \\ users : \mathbb{F}\ UserId \\ playlists : PlaylistId \pfun Playlist \\ playlist\_owner : PlaylistId \pfun UserId \\ playlist\_subscribers : PlaylistId \pfun \mathbb{F}_1\ UserId \\ \hline \vdots \end{array}$$

Complete the definition by capturing the following constraints.

(a) Only songs that appear in *songs* can appear in *playlists*.

(b) Only playlists that appear in *playlists* can appear in *playlist_owner*.

(c) Only users that appear in *users* can appear in *playlist_owner*.

(d) Only playlists that appear in *playlists* can appear in *playlist_subscribers*.

(e) Only users that appear in *users* can appear in *playlist_subscribers*.

(f) Every playlist's owner is, by default, a subscriber to that playlist.

Exercise 49

Users can *love* or *hate* songs. This information is captured via the following.

$$\begin{array}{|l} hated : UserId \pfun \mathbb{F}\, SongId \\ loved : UserId \pfun \mathbb{F}\, SongId \\ \hline \vdots \end{array}$$

Complete the definition by capturing the following constraints.

(a) Only users that appear in *users* can appear in *hated*.

(b) Only songs that appear in *songs* can appear in *hated*.

(c) Only users that appear in *users* can appear in *loved*.

(d) Only songs that appear in *songs* can appear in *loved*.

(e) No user can both hate and love the same song.

Exercise 50

Show how the following — all of which rely on the definitions of the previous two questions — can be represented via the mathematical language of Z.

(a) The users who have subscribed to no playlists.

(b) The playlists with at least 100 subscribers.

(c) The user who loves the most songs. (You should feel free to assume that there is a unique such user.)

(d) The song that is loved by the most users. (Again, you should feel free to assume that there is a unique such song.)

Exercise 51

Let's now consider two further functions. The first, *length*, maps a given song (identifier) to its length (in seconds). The second, *popularity*, assigns a popularity score to each song (identifier).

$$\begin{array}{|l} length : SongId \pfun \mathbb{N} \\ popularity : SongId \pfun \mathbb{N} \\ \hline \vdots \end{array}$$

(a) Complete the above definition, so that:

- songs appearing in *length* and *popularity* must appear in *songs*; and
- the *popularity* of a score is determined by the difference between the number of people who love it and the number of people who hate it — with a non-positive score capped at 0 — multiplied by the number of playlists in which it appears.

So, for example, if a song is liked by 6 people, hated by 3 and appears in 10 playlists, then its popularity score will be 30. If, on the other hand, a second song is liked by 3 people, hated by 6 people and also appears in 10 playlists, then its popularity score will be 0.

(b) Define, via an axiomatic definition, the most popular song that appears in *songs*. If there is not a unique such element, the function should return *null_song*, which is an element of *SongId*.

(c) Define, via a set comprehension, the set of playlists that feature the most popular song that appears in *songs*. You should assume that *null_song* appears in no playlists.

Exercise 52

We now consider users playing songs. To do this, we assume the existence of timestamps:

$$Tstamp == \mathbb{N}$$

We also assume the existence of a free type *Status*:

$$Status ::= premium \mid standard$$

and a function *user_status*:

$$user\_status : UserId \pfun Status$$

$$\vdots$$

(For the sake of simplicity, we shall assume, for now, that a user's status is 'fixed', i.e. they can't move from one status to another.)

The instance of a song being played is captured thus:

$$Play == SongId \times UserId \times Tstamp$$

(a) Define a recursive function, *premium_plays*, that takes a sequence of type seq *Play* and filters it with respect to those songs played by users whose status is *premium*.

(b) Define a recursive function, *standard_plays*, that takes a sequence of type seq *Play* and filters it with respect to those songs played by users whose status is *standard*.

(c) Define a recursive function, *cumulative_length*, that takes a sequence of type seq *Play* and returns the cumulative length of the songs in p.

And one final modelling question ...

A movie company has approached you to construct a model to help it keep track of a new series of films.

These films feature *characters*, who, at any point, can be *alive* or *dead*. There are three types of character: superheroes, immortals, and humans. Some superheroes, following death, can, at some future point, become alive again; immortals never die; humans live, then die. Each superhero has a specific limit on how many lives they can have (drawn from $1 \ldots 3$); the life limit for immortals and humans is 1.

What complicates matter is that the company needs to keep track of characters across several *timelines*. It may be that a character exists in only one timeline; it may be that they exist in more than one timeline; it may be that they are alive in one timeline and dead in another. A collection of timelines is called a *multiverse*.

Unfortunately, the notion of a multiverse is considered rather passé by the company. As such, they have asked you to construct a model that keeps track of a *megaverse* (i.e. a multiverse of multiverses). For the sake of clarity: a megaverse is a set of sets of timelines. So, for example, the megaverse might consist of two multiverses (say, $M1$ and $M2$), with multiverse $M1$ consisting of timelines $T1$ and $T2$ and multiverse $M2$ consisting of timelines $T3$ and $T4$. To help keep track of things, you should assume the existence of timeline ids and multiverse ids. Each such identifier can appear no more than once in the megaverse. Furthermore, for reasons that cannot be explained by the movie company, each non-empty multiverse must have a unique number of timelines.

There are two things to note with respect to characters. First, a human can become a superhero — but can only do so while they are alive. At this point, their 'life limit' may change (from 1 to either 2 or 3). Second, an immortal can become human (and, consequently, can die). It follows that a character may move from immortal to human to superhero (but there is no movement in the other direction). Importantly, such changes only happen within timelines — so, for example, a character may be an immortal in one timeline, a human in a second timeline, and a superhero in a third.

(a) Define appropriate entities to keep track of this scenario.

(b) Define a function that takes as input a timeline id and returns as output all characters associated with that timeline, together with their current status in a (name, character type, alive or dead) triple.

(c) Define a function that takes as input a character and returns as output the current status — in terms of a (timeline id, character type, alive or dead) triple — of that character across all timelines.