University of Portsmouth
BSc (Hons) Computer Science
Second Year

# Discrete Mathematics and Functional Programming (DMAFP)

M21274
January 2024 - June 2024
20 Credits

Thomas Boxall
up2108121@myport.ac.uk

# Contents

# Part I

# Discrete Maths

# Page 1

# Lecture - Sets

2024-01-23 17:00 Janka

## 1.1 Introduction

*Sets* underpin maths and Computer Science. A set is a collection of objects, which are called the elements (also known as members of the set). For example, a set of the numbers 1, 3, 8; or the collection of students in a class born in March. There are two characteristics of sets:

1. There are no repeated occurrences of elements
2. There is no particular order of the elements

## 1.2 Set Notation

The elements of a set are enclosed in braces with their names being denoted by a *letter*, for example:

$$A = \{1, 2, 3\}, \quad C = \{Portsmouth, Brighton, London\}$$

There are two ways that we can describe the members of a set. We can *list the elements* which is mainly used for finite sets, for example:

$$A = \{3, 6, 9, 12\}$$

We can *specify a property* that all the elements in the set have in common. The '|' character is read 'such that', sometimes ':' is used in it's place. For example:

$$B = \{x | x \text{ is a multiple of 3 and } 0 < x < 15\}$$

We can also use *three dots* to informally denote a sequence of elements that we don't wish to write down, for example:

$$C = \{1, \ldots, 10\}$$

### 1.2.1 Sets of Numbers

There are some reserved letters to denote specific sets of numbers in maths. These are shown below:

- $\mathbb{N}$ (or $N$) is used for the set of natural numbers (integers >= 0). $\mathbb{N} = \{0, 1, 2, 3, 4, \ldots\}$
- $\mathbb{Z}$ (or $Z$) is used for the set of integers. $\mathbb{N} = \{\ldots, -1, 0, 1, \ldots\}$
- $\mathbb{Q}$ (or $Q$) is used for the set of rational numbers (number which can be expressed as a quotient or fraction). $\mathbb{Q} = \{0, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots\}$
- $\mathbb{R}$ (or $R$) is used for the set of real numbers. $\mathbb{R} = \{\ldots, -1, 0, \frac{1}{2}, \ldots\}$

### 1.2.2 Elements of a Set

We can use the $\in$ symbol to denote if an element is a member of a given set. For example, if $x$ is a member of $S$ - then we can say:

$$x \in S$$

The symbol $\notin$ denotes an element is not a member of a given set. For example, if $y$ is **not** a member of $S$ - then we can say:

$$y \notin S$$

### 1.2.3 Many Ways to Say The Same Thing

There are several ways of describing the same set, for example for the set $S$ of *odd integers*:

$$\begin{aligned} S &= \{\ldots, -5, -3, -1, 1, 3, 5, \ldots\} \\ &= \{x | x \text{ is an odd integer }\} \\ &= \{x | x = 2k + 1 \text{ for some integer } k\} \\ &= \{x | x = 2k + 1 \text{ for some } k \in \mathbb{Z}\} \\ &= \{2k + 1 | k \in \mathbb{Z}\} \end{aligned}$$

The phrase "for some [integers $K$]", means "for all [integers $k$]"

### 1.2.4 Empty Sets

Where a set has *no elements*, it is called an empty set or null set. It's denoted with the $\emptyset$ symbol, for example:

$$\emptyset = \{\}$$

### 1.2.5 Finite & Infinite Sets

If the number of elements in the set is fixed (for example when counting the elements at a fixed rate for a set amount of time), then the set is *finite*. If the set $X$ is finite, then we call $|X|$ the *cardinality* of $X$ therefore:

$$|X| = \text{number of elements in } X$$

If the counting never stops then $X$ is an infinite set.

### 1.2.6 Subsets

A subset is where one set's elements are entirely present in another set. There are three conditions we need to know about:

- $A \subseteq B$: $A$ is a subset of $B$ therefore every element in $A$ is also in $B$.
- $A \nsubseteq B$: $A$ is not a subset of $B$.
- $A \subset B$: $A$ is a proper subset of $B$, therefore $B$ has at least one additional element which is not in $A$.

### 1.2.7 Equality of Sets

Two sets are *equal* if they have exactly the same elements. This is denoted by writing $A = B$. Where $A = B$, the following conditions are also true:

- $A \subseteq B$ for every $a$ if $a \in A$, then $a \in B$
- $B \subseteq A$ for every $b$ if $b \in B$, then $b \in A$

## 1.3 Operations on Sets

Sets can have *operations* performed on them - this will change something about them.

### 1.3.1 Intersection

The intersection of two sets $A$ and $B$ is defined as:

$$A \cap B = \{x|x \in A \text{ and } x \in B\}$$

This is the set of elements which appear in both sets only. If we take a Venn Diagram with a set on either side - its the overlapped elements which would be returned from an intersection operation. For example if $A = \{a, b, c\}$ and $B = \{c, d\}$ then $A \cap B = \{c\}$.

### 1.3.2 Disjoint

If an intersection returns no elements, then the two sets are *disjoint*. This is shown by:

$$A \cap B = \emptyset$$

### 1.3.3 Union

The *union* of the two sets $A$ and $B$ is defined as:

$$A \cup B = \{x|x \in A \text{ or } x \in B\}$$

This is the set of elements which are in either $A$ or $B$, this means elements which appear in both are returned. For example, if $A = \{a, b, c\}$ and $B = \{c, d\}$ then $A \cup B = \{a, b, c, d\}$.

### 1.3.4 Difference

The *difference* between two sets, $A$ and $B$ is defined as:

$$A \setminus B = \{x|x \in A \text{ or } x \in B\}$$

This is the set of elements which are in $A$ but not in $B$, so could be represented as $A - B$. Note that $A \setminus B \neq B \setminus A$.

### 1.3.5 Counting Elements In a Set

If we take $A$ and $B$ to be finite sets, we can calculate the number of elements in the union of $A$ and $B$. The correct way to count this is as follows:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

We have to minus $|A \cap B|$ from the sum because otherwise it is as though we are counting it twice due to the fact that we are summing the total number of elements in $A$ and $B$.

## 1.4 Complement

If we consider that all subsets are the subset of a particular set, $U$ for example (the universe of discourse), then the difference $U \setminus A$ is called the *complement* of $A$ is shown as either $\overline{A}$ or $A'$. For example:

$$A' = \{X|x \in U \text{ and } x \notin A\}$$

## 1.5 Basic Set Properties

Sets have a number of basic properties - many of these are the same as that for Boolean Expressions

- $A \cup \emptyset = A$
- $A \cap \emptyset = \emptyset$
- $A \cup A = A$
- $A \cap A = A$
- Commutative
  - $A \cup B = B \cup A$
  - $A \cap B = B \cap A$
- Associative
  - $(A \cup B) \cup C = A \cup (B \cup C)$
  - $(A \cap B) \cap C = A \cap (B \cap C)$
- Distributive
  - $A \cap (B \cap C) = (A \cap B) \cup (A \cap C)$
  - $A \cup (B \cup C) = (A \cup B) \cap (A \cup C)$
- de Morgan's
  - $(A \cap B)' = A' \cup B'$
  - $(A \cup B)' = A' \cap B'$

## 1.6 Power Set

A *power set* is the collection of all subsets of a set, $S$ which is denoted by $P(S)$. For example, if $S = \{a, b, c\}$ then:

$$P(S) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$$

## 1.7 Partition

A *partition* of the set $S$ is a collection of non-empty subsets of set $S$ where every element form $S$ belongs to exactly one member of $S$. This means that the sets are mutually disjoint and that the union of all the sets in the collection results in the original set, $S$. For example, if $S = \{a, b, c, d, e, f\}$ then $\{\{a, e\}, \{c\}, \{f, d\}, \{b\}\}$ is a partition of $S$.

# Page 2

# Lecture - Relations

2024-01-30 17:00 Janka

## 2.1 Ordered Pairs

An ordered pair of elements is a group of two elements which are in a specific order. They are written as $(a, b)$ and the order matters - this means $(a, b)$ is distinct from the pair $(b, a)$. Note that ordered pairs use the brackets () while sets use curly braces {}

## 2.2 Cartesian Product

The Cartesian Product of two sets is the set of **all** ordered pairs where the first element is taken from set 1 and the second element from set 2. The formal definition is as follows:

$$A \times B = \{(a, b) | a \in A \text{ and } b \in B\}$$

For example - if $X = \{1, 2, 3\}$ and $Y = \{a, b\}$, then

$$X \times Y = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$$

## 2.3 Relations

A relation is the set of subsets from a cartesian product. For example, if we take $A = \{a, b, c, d, e\}$ and $B = 1, 2, 3$ then:

$$R_1 = \{(a, 1), (b, 1), (c, 2), (c, 3)\}$$
$$R_2 = \{(a, 3), (a, 1), (c, 2), (c, 1), (b, 2)\}$$

$R_1$ and $R_2$ are both examples of Binary relations from A to B.

### 2.3.1 Describing Relations

To describe a relation, we could list all of its elements, however this can be very long and obtuse so it's better & more common practice to use "the characteristics of their elements".

### 2.3.2 Relations On A Set

A relation *on a set* is where both sets are equal. For example $A = B$ then a relation on $A$ is a relation from $A$ to $A$, hence a subset of $A \times A$.

For example, let $R$ be the relation on $A = \{1, 2, 3, 4\}$ as defined by:

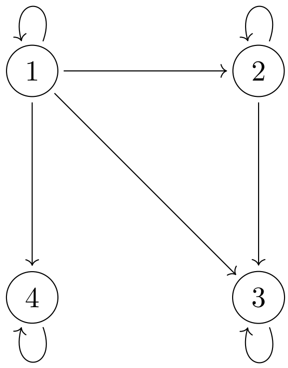
$$(x, y) \in R \text{ if and only if } x \text{ divides } y, \text{ for all } x, y \in A$$

Then we can conclude that $R$ is:

$$R = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4)\}$$

## 2.4 A Digraph

We can use a *digraph* to picture a relation on a set. An example is shown below:




The dots (vertices) represents the elements of $A = \{1, 2, 3, 4\}$. If the element $(x, y)$ is in the relation, an arrow (directed edge) is drawn from $x$ to $y$.

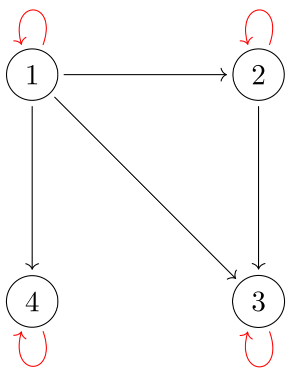
$$R = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4)\}$$

### 2.4.1 Reflexivity

A relation is reflexive where for all elements in the set - there is an ordered pair in which both elements are the same, for example $(1, 1)$ or $(2, 2)$. In reference to the set $A = \{1, 2, 3, 4\}$, the relation:

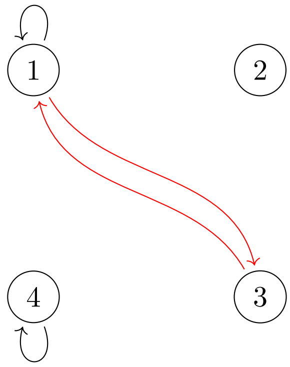
$$R = (1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4)$$

On the following digraph, the red arrows are the ones which display the reflexivity.



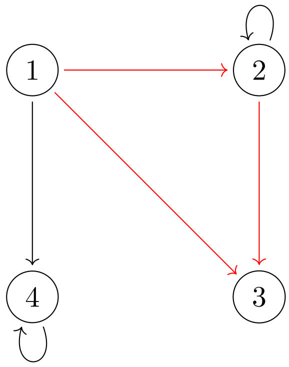

### 2.4.2 Symmetry

A relation is symmetrical where $(x, y) \in R$ and $(y, x) \in R$. If the condition is not true, then we do not have symmetry.




### 2.4.3 Transitivity

For a binary relation, $R$ on set $A$; $R$ is transitive if and only if for all $x, y, z \in A$ if $(x, y) \in R$ and $(y, z) \in R$ and $(x, z) \in R$. We initially assume that a relation is transitive and try to disprove it; if we are unable to disprove it then the relation is transitive. In the event that there is only one element in the relation - the relation will *always* be transitive.




### 2.4.4 Equivalence

Where a relation is reflexive, symmetric and transitive - it is classed as an equivalence.

#### 2.4.4.1 Equivalence Class

To be continued.

# Page 3

# Lecture - Functions

2024-02-06 17:00 Janka

A *function* can be described in two ways. The mathematical definition is that “a function is a special type of relation in which a single input will have at most one output”. The alternative definition of a function is that it is a mysterious black box which takes an input and returns and output. The same function, with the same input will always return the same output.

If we take $A$ and $B$ to be nonempty sets, then: $A$ is a (total) function $f$ from $A$ to $B$, $f : A \to B$, is a relation from $A$ to $B$ such that

$$\text{for all } x \in A \text{ there is exactly on element in } B, f(x)$$

associated with $x$ by relation $f$. Note that the word ‘total’ is used to describe the above function which means that every input has a defined output. The function $f : \mathbb{Z} \to \mathbb{Z}$ which is defined by $f(x) = 2x$ is also an example of a total function.

It is also possible to have a ‘partial’ function, this is where some of the inputs do not have defined outputs. For example the function $f(\frac{1}{x})$ where $x = 0$, would be undefined therefore the function is classed as ‘partial’.

## 3.1 Describing A Function

There are a few different ways in which a function can be described.

### 3.1.1 By A Formula

This is the most common method used. The function $f$ from $\mathbb{N} \to \mathbb{N}$ that maps every natural number $x$ to its cube $x^3$ can be described as:

$$f(x) = x^3$$

### 3.1.2 By All Possible Associations

Whilst this is a valid method, it will generally not be used for efficiency reasons. The function $g$ from $A = \{a, b, c\}$ to $B = \{1, 2, 3\}$ would be shown as:

$$g(a) = 1, g(b) = 1, g(c) = 2$$

## 3.2 Domain, Co-Domain & Range

The domain of a function is the set of all input values for which there is a defined output. For example if we let $f : A \to B$ then the subset $D \subset A$ of all elements for which $f$ is defined is the domain. In the case of a total function, $D = A$ and in the case of a partial function, $D \subseteq A$

The co-domain of a function is the set of all possible output values; not just the ones which map to an input. For example, if we let $f : A \to B$ then the set $B$ is the co-domain.

The range (also sometimes known as the image) of a function is the set of elements in the co-domain which map to an input. For example, if we let $f : A \to B$ then the range is denoted by $range(f)$. The range can also be expressed as:

$$range(f) = \{f(x)|x \in A\}$$

## 3.3 Properties of Functions

Functions have a number of properties.

### 3.3.1 Injective

The function $f : A \to B$ is injective (or one-to-one) if there is only one input that maps to each output. It can mathematically be defined as:

$$\text{for all } x, y \in A \text{ if } x \neq y \Rightarrow f(x) \neq\Rightarrow f(y)$$

### 3.3.2 Surjective

A function $f : A \to B$ is surjective (or onto) if the $range(f)$ is the co-domain $B$. It can mathematically be defined as

$$\text{for all } y \in B \text{ there exists } x \in A \text{ such that } f(x) = y$$

A function which is not *onto* is *into*.

### 3.3.3 Bijective

A function $f : A \to B$ is bijective (or one-to-one correspondence) if it is both injective and surjective.

## 3.4 Composite Functions

A new function can be constructed by combining other simpler functions in some way. If we let $f : A \to B$ and $gB \to C$ be functions. The composition of $g$ with $f$ is the function denoted by $g \circ f : A \to C$ and defined by:

$$(g \circ f)(x) = g(f(x)) \text{ for all } x \in A$$

$(g \circ f)(x) = g(f(x))$ is read as $g$ of $f$, which means do $f$ first then do $g$.

## 3.5 Inverse Functions

An inverse function is where the output of function $f$ can be fed into the input of function $f^{-1}$ to get the original input of $f$. This is mathematically defined as: $f : X \to Y$ is a bijective function, then there is an inverse function $f^{-1} : X \to Y$ that is defined as:

$$f^{-1}(y) = x \text{ if and only if } f(x) = y$$

# Part II

# Functional Programming

# Page 4

# Lecture - Introduction to Functional Programming

2024-01-22 1200 Matthew

## 4.1 Introduction

*Functional Programming* is a different programming paradigm. There are all sorts of different ways we can classify a programming language, paradigm being one of them. More details on this in another module.

### 4.1.1 Imperative vs Functional Programming

Before we go too deep into Functional Programming, we will first look at the structure of Imperative Programming.

Imperative Programming is a paradigm where the execution of the program consists of executions of *statements*, which each impact the program's *state*. *Side Effects* can be caused by the statements; these are things that the program does where it cannot guarantee the outcome - for example get the current temperature, ask the user to enter a number or getting the system time.

Pure Functional Programming does not have a state, does not have statements and does not have side effects. However - side effects are a "necessary evil" so they get brought back in isolated from the main program. Once side effects are introduced, our functional programming becomes impure.

## 4.2 Functional Programming

In Functional Programming, there are three key terms - expression, evaluation and value. An expression is some text which has a value, for example `2 * 3 + 1`; a value is the thing which the expression has, for example `7`; and evaluation is the process used to obtain a value from an expression.

We will start our FunProg journey looking at Mathematical Functions, which we can think of as a box which maps argument values to a result value. A Haskell program is mainly made up of Function definitions, for example

```
square :: Int -> Int
square n = n * n
```

The first line above starts with the function name, then lists the parameters (a single `Int`), then finally the result type. The second line is the actual function, starting with the name, then the names assigned to the parameters, then the value which gets returned. The double colon (`::`) is read as 'is type of'.

Haskell comes with basic data types. This includes the `Bool` which can either be `True` or `False` and can have operations performed on it through the Boolean Operators. Or is implemented as *exclusive or* where it gives True when exactly one of it's arguments is true. Haskell also includes the `Int` and `Float` datatypes. These come with the standard functions and orders of precedence rules you would expect.

Haskell also has conditional expressions which takes the form:

```
if condition then m else n
```

More information will be provided in the coming lectures as to alternatives to the if expression.