Discrete Mathematics

	topic
week 1	***Sets***
week 2	***Functions***
week 3	***Counting***
week 4	***Relations***
week 5	***Sequences***
week 6	***Modular Arithmetic***
week 7	***Asymptotic Notation***
week 8	***Orders***

Jonathan Barrett
jonathan.barrett@cs.ox.ac.uk

Material by Andrew Ker
University of Oxford
Department of Computer Science

Discrete Mathematics

Jonathan Barrett

jonathan.barrett@cs.ox.ac.uk

Material by Andrew Ker

University of Oxford

Department of Computer Science

Chapter 2: Functions

Intervals

An **interval** is a subset I of $\mathbb{R}$ with the **interval property**:

$$x, z \in I \text{ and } x < y < z \;\Rightarrow\; y \in I$$

Intervals have a concise notation:

$$\begin{aligned}
(a, b) &= \{x \in \mathbb{R} \mid a < x < b\} \\
(a, b] &= \{x \in \mathbb{R} \mid a < x \le b\} \\
(a, \infty) &= \{x \in \mathbb{R} \mid a < x\} \\
(-\infty, b) &= \{x \in \mathbb{R} \mid x < b\}
\end{aligned}
\qquad
\begin{aligned}
[a, b] &= \{x \in \mathbb{R} \mid a \le x \le b\} \\
[a, b) &= \{x \in \mathbb{R} \mid a \le x < b\} \\
[a, \infty) &= \{x \in \mathbb{R} \mid a \le x\} \\
(-\infty, b] &= \{x \in \mathbb{R} \mid x \le b\}
\end{aligned}$$

Intervals

An **interval** is a subset I of $\mathbb{R}$ with the **interval property**:

$$x, z \in I \text{ and } x < y < z \;\Rightarrow\; y \in I$$

Intervals have a concise notation:

$$\begin{aligned}
(a,b) &= \{x \in \mathbb{R} \mid a < x < b\} \\
(a,b] &= \{x \in \mathbb{R} \mid a < x \leq b\} \\
(a,\infty) &= \{x \in \mathbb{R} \mid a < x\} \\
(-\infty,b) &= \{x \in \mathbb{R} \mid x < b\}
\end{aligned}$$

$$\begin{aligned}
[a,b] &= \{x \in \mathbb{R} \mid a \leq x \leq b\} \\
[a,b) &= \{x \in \mathbb{R} \mid a \leq x < b\} \\
[a,\infty) &= \{x \in \mathbb{R} \mid a \leq x\} \\
(-\infty,b] &= \{x \in \mathbb{R} \mid x \leq b\}
\end{aligned}$$

- Intervals which don't include their endpoints are called open intervals.
- Intervals which do include all their endpoints are called closed intervals.
- Intervals [a,b) and (a,b] are called half-open intervals.

Functions

A **function** associates elements of one set with another. It consists of:

- A set A called the **domain**,
- A set B called the **codomain**,
- A **map** which associates exactly one element of B with each element of A.

We write

$$f : A \to B$$

to indicate that f is a function with domain A and codomain B, and

$$f : a \mapsto b \quad \text{or} \quad f(a) = b$$

to indicate that f associates b with a.

Functions

A **function** associates elements of one set with another. It consists of:

- A set A called the **domain**,
- A set B called the **codomain**,
- A **map** which associates exactly one element of B with each element of A.

We write

$$f : A \to B$$

to indicate that f is a function with domain A and codomain B, and

$$f : a \mapsto b \quad \text{or} \quad f(a) = b$$

to indicate that f associates b with a.

Formally, functions have one "input" and one "output". Multiple inputs or outputs correspond to A or B being a cartesian product.

Equality of Functions

Two functions are equal only if all three components — domain, codomain, map — are **all** the same.

If $f\colon A \to B$ and $g\colon A' \to B'$ then $f = g$ only when

- $A = A'$,
- $B = B'$,
- $f(a) = g(a)$ for all $a \in A$.

Partial Functions

Sometimes we want to place a looser condition on the inputs and outputs.

A **partial function** associates elements of one set with another. It consists of:

- A set A called the **domain**,
- A set B called the **codomain**,
- A **map** which associates exactly **zero or one** element of B with each element of A.

Roughly speaking: a partial function may be "not defined" on some of its inputs.

When we want to emphasise that a function is not partial we call it a **total function**.

Properties of Functions

Let f be a function, $f\colon A \to B$. We denote the domain of a function f by $\mathrm{Dom}(f)$.

We write $\mathrm{Im}(f) = \{b \in B \mid f(a) = b \text{ for some } a \in A\}$.

This is the **image** of f (also known as the **image of A under f**).

Properties of Functions

Let f be a function, $f\colon A \to B$. We denote the domain of a function f by $\mathrm{Dom}(f)$.

We write $\mathrm{Im}(f) = \{b \in B \mid f(a) = b \text{ for some } a \in A\}$.

This is the **image** of f (also known as the **image of A under** f).

f is **onto** if every element of B is associated with some element of A, i.e.

$$\mathrm{Im}(f) = B.$$

f is **1-1** if no element of B is associated with more than one element of A, i.e.

$$a_1 \neq a_2 \Rightarrow f(a_1) \neq f(a_2) \text{ for all } a_1, a_2 \in A.$$

f is **bijective** if it is both onto and 1-1.

Properties of Functions

Let f be a function, $f\colon A \to B$. We denote the domain of a function f by $\mathrm{Dom}(f)$.

We write $\mathrm{Im}(f) = \{b \in B \mid f(a) = b \text{ for some } a \in A\}$.

This is the **image** of f (also known as the **image of A under f**).

f is **onto** if every element of B is associated with some element of A, i.e.

$$\mathrm{Im}(f) = B.$$

f is **1-1** if no element of B is associated with more than one element of A, i.e.

$$a_1 \neq a_2 \Rightarrow f(a_1) \neq f(a_2) \text{ for all } a_1, a_2 \in A.$$

f is **bijective** if it is both onto and 1-1.

Some books use other terminology:

- ***instead of "1-1" they say "injective"; a 1-1 function is an "injection";***
- ***instead of "onto" they say "surjective"; an onto function is a "surjection";***
- ***a bijective function is called a "bijection".***

Bijections

A bijection matches up elements of A and B.

So if there is a bijection $f\colon A \to B$ then

$$|A| = |B|.$$

(In fact the converse is also true: there always exists a bijection between sets of equal cardinality.)

Whenever there is a bijection $f\colon A \to B$ there is always another bijection $g\colon B \to A$, so sometimes we say that f is a bijection **between** A **and** B.

Proof of the Contrapositive

The statement

$$P \Rightarrow Q$$

means "if P then Q" ("P implies Q"). The statement

$$\neg P$$

means "P is false" ("not P"). It is a fact that

$$P \Rightarrow Q \;\text{ and }\; \neg Q \Rightarrow \neg P$$

are logically equivalent.

(Remember that $P \Rightarrow Q$ ***and*** $Q \Rightarrow P$ ***are NOT logically equivalent.)***

If we want to prove $P \Rightarrow Q$ it is sometimes easier to prove $\neg Q \Rightarrow \neg P$ instead. This is called "proof of the contrapositive".

Proof by Contradiction

Suppose we want to prove P.

Claim P

Proof Assume $\neg P$ for a contradiction

therefore …

therefore …

therefore …

therefore … (something impossible or contradictory) … $\#$

Therefore P. $\square$

Functional Composition

If $f\colon A \to B$ and $g\colon B \to C$ then it makes sense to put the "output" of f as an "input" of g. This is called the **composition** of g with f.

$$(g \circ f) : A \to C, \quad (g \circ f)(x) = g(f(x))$$

$g \circ f$ is pronounced "g after f".

Functional Composition

If $f\colon A \to B$ and $g\colon B \to C$ then it makes sense to put the "output" of f as an "input" of g. This is called the **composition** of g with f.

$$(g \circ f) : A \to C, \quad (g \circ f)(x) = g(f(x))$$

$g \circ f$ is pronounced "g after f".

Note that

- *The composition $g \circ f$ does not exist if $\mathrm{Dom}(g)$ does not match the codomain of f.*
- *The order of composition matters: in general, $g \circ f \neq f \circ g$.*

Functional Composition

If $f : A \to B$ and $g : B \to C$ then it makes sense to put the "output" of f as an "input" of g. This is called the **composition** of g with f.

$$(g \circ f) : A \to C, \quad (g \circ f)(x) = g(f(x))$$

$g \circ f$ is pronounced "g after f".

Note that

- ***The composition $g \circ f$ does not exist if $\mathrm{Dom}(g)$ does not match the codomain of f.***
- ***The order of composition matters: in general, $g \circ f \neq f \circ g$.***

<u>Claim</u> Composition of functions is associative

i.e. if $f : A \to B$, $g : B \to C$, $h : C \to D$

then

$$h \circ (g \circ f) = (h \circ g) \circ f.$$

Inverse Function

If $f\colon A \to B$ and $g\colon B \to A$ satisfy both

$$g \circ f = \mathrm{id}_A \qquad g(f(a)) = a \text{ for all } a \in A$$

$$f \circ g = \mathrm{id}_B \qquad f(g(b)) = b \text{ for all } b \in B$$

then g is the **inverse** function to f, and we write

$$g = f^{-1}.$$

Inverse Function

If $f\colon A \to B$ and $g\colon B \to A$ satisfy both

$$g \circ f = \mathrm{id}_A \qquad g(f(a)) = a \text{ for all } a \in A$$

$$f \circ g = \mathrm{id}_B \qquad f(g(b)) = b \text{ for all } b \in B$$

then g is the **inverse** function to f, and we write

$$g = f^{-1}.$$

Fact f has an inverse if and only if f is bijective.

Restricted Function

If $f: A \to B$ and $A' \subseteq A$ then we can reduce f to a function with domain A'

$$f\restriction_{A'}: A' \to B, \quad f'(a) = f(a) \text{ for } a \in A'.$$

Fact If i is the **inclusion map** $i: A' \to A$, $i(a) = a$, then

$$f\restriction_{A'} = f \circ i.$$

Diversion: Binary Operators

A function

$$f : A \times A \to A$$

is called a **binary operator** on A.

They are often written infix:
$$\begin{aligned} + : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, &\quad x + y = \ldots \\ \cup : \mathcal{S} \times \mathcal{S} \to \mathcal{S}, &\quad A \cup B = \ldots \end{aligned}$$

Diversion: Binary Operators

A function

$$f : A \times A \to A$$

is called a **binary operator** on A.

They are often written infix:

$$+ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, \quad x + y = \ldots$$
$$\cup : \mathcal{S} \times \mathcal{S} \to \mathcal{S}, \quad A \cup B = \ldots$$

A binary operator . on A

- is **idempotent** if $x \cdot x = x$ for all $x \in A$
- is **commutative** if $x \cdot y = y \cdot x$ for all $x, y \in A$
- is **associative** if $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ for all $x, y, z \in A$
- has an **identity element** e if $e \cdot x = x \cdot e = x$ for all $x \in A$.

An identity element is sometimes called a **zero**, **one** or **unit**. A set together with a binary operator which is associative and has an identity is called a **monoid**.

Discrete Mathematics

Jonathan Barrett
`jonathan.barrett@cs.ox.ac.uk`
Material by Andrew Ker
University of Oxford
Department of Computer Science

End of Chapter 2