Mathematical Techniques for Computer Science

Todd Davies

June 1, 2016

Introduction

This course covers the fundamental maths required by Computer Science students in order to successfully complete the reminder of their courses as well as for a career in computer science. It includes modules on discrete structures, set theory, logic, probability, mathematical induction, relations, vectors, matrices and transformation.

Aims

This full-year course unit focuses on the use of mathematics as a tool to model and analyse real-world problems arising in computer science. Four principal topics, drawn from the traditional areas of discrete mathematics as well as some continuous mathematics, will be introduced: symbolic logic, probability, discrete structures, and vectors and matrices. Each topic will be motivated by experts introducing relevant real-world problems arising in their own specialism.

Abstraction is fundamental to computer science. Hence, a fundamental emphasis of this course unit is to introduce mathematical techniques and skills to enable the student to design and manipulate tractable and innovative abstract models of chunks of the real-world. These techniques and skills include appropriate mathematical notations and concepts. These range over the four principal areas mentioned above. Formalisation in mathematics has, in general, significant cost. Therefore, to be of practical use, the benefits arising from formalisation, such as succinctness, unambiguity, provability, transformability and mechanisability, must outweigh the costs. A key aim of the course is for the student to appreciate this issue and know how and when to use particular techniques.

The specific aims of the course unit are:

- To demonstrate the relevance of mathematics to computer science.
- To introduce fundamental mathematical techniques of abstraction.
- To demonstrate applicability of particular mathematical techniques and skills for particular types of computer science problem.
- To appreciate the costs and benefits of mathematical modelling.
- The delivery style will place more emphasis on students undertaking appropriate background reading, i.e. being more independent learners, and use the lectures more to demonstrate examples and solutions and not working through every detail of a particular or concept.

The course unit is delivered by staff from both the School of Computer Science and the School of Mathematics.

Licence and contribution

Contents

1 Functions

A mathematical function is a relation between a set of inputs and a set of outputs. Each input is related to exactly one output, and the same input will *always* result in the same output.

1.1 Simple functions

Functions can be split into three components; a source, a target and the behaviour of the function.

The source and target of a function are both sets, and are denoted as S and T respectively in the following notation:

$$f : S \to T$$

In order to find the value of a function f for a specific input x, the notation is:

$$f(x)$$

Describing the behaviour of a function requires another notation:

$$x \longmapsto \ldots x \ldots$$

Consequently, to completely describe a function by defining it's source, target and behaviour, we must write something like:

$$\begin{aligned} f : S &\to T \\ x &\longmapsto \ldots x \ldots \end{aligned}$$

An example of this may be a function that finds the absolute value of integers:

$$\begin{aligned} abs : \mathbb{Z} &\to \mathbb{N} \\ x &\longmapsto |x| \end{aligned}$$

1.2 Equality of functions

Two functions are said to be equal to each other if the following conditions are met:

- The source set is the same
- The target set is the same
- The behaviour of the functions is the same (so for every $x \in S$, where S is the source set, both functions give the same output y when given x as the input, where $y \in T$ and T is the target set).

1.3 Identity functions

For any set S, there is a function such that:

$$\begin{aligned} f : S &\to S \\ x &\longmapsto x \end{aligned}$$

This is described as the identity function of S, and is often denoted by 1_S or id_S

1.4 Injectivity, Surjectivity and Bijectivity

The terms *injective*, *surjective* and *bijective* are used to describe which elements in a functions source set are mapped to which elements it's target set:

Injective
For each $x \in S$ there is at most one $y \in T$ such that $f(x) = y$.

Surjective
For each $x \in S$ there is at least one $y \in T$ such that $f(x) = y$.

Bijective
For each $x \in S$ there is only one $y \in T$ such that $f(x) = y$.

Note that if an function is injective and surjective, then it must also be bijective.

1.4.1 Permutation of sets

A function is said to be a permutation of a set if the source and target of the function is the same set and the function is bijective. Applying the function effectively re-arranges the members of the set.

1.5 Function composition

Two functions can be composited when the source set of one function is the same as the target set of another. For example:

$$f : S \to T$$
$$g : T \to U$$

Here, we can compose a new function made up of f and g:

$$h = g \circ f$$

Or in other words:

$$h(x) = g(f(x))$$

When we're invoking composition on more than two functions, it is an associative operation. For this reason, we often omit brackets and the circle operator:

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

Functions with recursive definitions

If a function references itself in its own body, then it is recursive. An example might be:

$$f : \mathbb{N} \to \mathbb{N}$$
$$f(0) = 0$$
$$f(r + 1) = f(r) + 1$$

This function just returns the input it is given, but the

2 Induction

Induction is a technique that allows us to use the property of infinite countability of natural numbers ($\mathbb{N}$) to prove theorems.

Induction works by defining a **base case**, where some iterative variable (usually n) is equal to 0, and then coming up with a **step case**, that says when you add 1 to n for any n, then the result is also true. Since natural numbers are infinitely countable, then the step case must hold for all n.

2.1 Induction example

The most basic example of induction is to prove something like:

$$\forall n \in \mathbb{N}$$
$$0 + 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

Base case:
When $n = 0$ both sides of the equation equal 0.

Step case:
In order to prove the step case, we should first assign names to both equations:

$$L(n) = 0 + 1 + 2 + \cdots + n$$
$$R(n) = \frac{n(n+1)}{2}$$

Now, if we assume that $L(n) = R(n)$ as it is in the base case, we now need to prove that $L(n+1) = R(n+1)$:

$$\begin{aligned}
L(n+1) &= L(n) + (n+1) \\
&= R(n) + (n+1) \\
&= \frac{n(n+1)}{2}(n+1) \\
&= \frac{n(n+1) + 2(n+1)}{2} \\
&= \frac{n(n+1) + 2n + 2)}{2} \\
&= \frac{n^2 + n + 2n + 2)}{2} \\
&= \frac{n^2 + 3n + 2)}{2} \\
&= \frac{n^2 + 3n + 2)}{2} \\
&= \frac{(n+2)(n+1)}{2} \\
&= R(n+1)
\end{aligned}$$

Summation notation

Often, you will be asked to provide an inductive proof for an equality where one side of the equation is a summation. The syntax is like so:

$$\sum_{i=0}^{n} i$$

This is the notation for adding all the numbers from 0 to n, like:

$$0 + 1 + 2 + \cdots + n$$

The number above the sigma is the limit of the summation, and the number below is the starting value.

Forming inductive proofs with this notation isn't much different to doing it without. All you do is add 1 to all the instances of your inductive variable and then rearrange the equation to get the other side.

An inductive proof of a summation

Lets prove that:

$$\sum_{i=0}^{n} x^i = \frac{x^{n-1} - 1}{x - 1}$$

So first, we have that:

$$\begin{aligned} L(n) &= \sum_{i=0}^{n} x^i \\ R(n) &= \frac{x^{n-1} - 1}{x - 1} \end{aligned}$$

First, lets prove the base case, when $n = 0$:

$$\begin{aligned} L(0) &= x^0 = 1 \\ R(0) &= \frac{-1}{-1} = 1 \end{aligned}$$

Now, we can prove the step case:

$$\begin{aligned} L(n+1) &= L(n) + x^{n+1} \\ &= R(n) + x^{n+1} \\ &= \frac{x^{n-1} - 1}{x - 1} + x^{n+1} \\ &= \frac{x^{n-1} - 1 + (x-1)(x^{n+1})}{x - 1} \\ &= \frac{x^{n-1} - 1 + x^n - x^{n-1}}{x - 1} \\ &= \frac{x^n - 1}{x - 1} \\ &= R(n+1) \end{aligned}$$

3 Relations

A relation is a set of pairs between two sets. You can think of it as a subset of the powerset of two sets. For example, if we have a relation R between the numbers $\{1, 2, 3\}$ and $\{4, 5, 6\}$, where the even and odd numbers are paired together, then it will look like this:

$$
\begin{aligned}
A &= \{1, 2, 3\} \\
B &= \{4, 5, 6\} \\
A \times B &= \left\{ \begin{matrix} (1,4) & (1,5) & (1,6) \\ (2,4) & (2,5) & (2,6) \\ (3,4) & (3,5) & (3,6) \end{matrix} \right\} \\
R \subset (A \times B) &= \left\{ \begin{matrix} (1,5) & \\ (2,4) & (2,6) \\ (3,5) & \end{matrix} \right\}
\end{aligned}
$$

If we want to represent that an element $a \in A$ is related to an element $b \in B$ via a relation R, then we write aRb.

Obviously, relations aren't related to natural numbers as in the previous example. You can have a relation between sets of any type. For example, you could define a relation between family members. The relation P could hold when an element a is the father of another element b.

If you can define a relation between family members, then it makes sense that you should also be able to draw a diagram like a family tree then. Such graphs are called *digraphs*, they might look like this:

$$
\begin{tikzcd}
 & bob \arrow[d] & \\
 & bill \arrow[dl] \arrow[dr] & \\
bilbo & & ben
\end{tikzcd}
$$

From this digraph, we can see that *bill R bilbo*, *bill R ben* and *bob R bill*.

3.1 Properties of relations

We can describe relations as possessing different properties; the four properties we cover in this course are as follows:

Reflexive:
A relation is reflexive if all the elements in the relation have a relation to themselves. In mathematical terms:

$$a \in A \textit{ then } aRa$$

Symmetric:
If a relation is symmetric, then if two elements are related, then the relation must go both ways. In other words:

$$(a, b) \in A \textit{ then } aRb \implies bRa$$

Antisymmetric:
A relation that is antisymmetric, then no relations work both ways. An example of an antisymmetric relation is the father relation we defined before; it is impossible for two distinct elements to be each others father. Mathematically,

$$(a, b) \in A \textit{ then } (aRb \wedge bRa) \implies a = b$$

Transitive:

A transitive relation is one where if an element a is related to another element b, and b is related to c, then a must be related to c:

$$(a, b, c) \in A \textit{ then } (aRb \wedge bRc) \implies aRc$$

3.2 Equivalence relation

A relation is said to be an equivalence relation if it is reflexive, symmetric and transitive. The idea is that if R is an equivalence relation, then if aRb, then in some sense, a and b are the same.

An equivalence relation splits a set into a number of partitions. Each partition is made up of elements of the set that are equivalent to each other. An *equivalence class* is a subset of elements in the set that are equivalent to each other as previously described. The notation for an equivalence class is $[a]_R$, where all the elements in that equivalence class will be related to a under the relation R.

Sometimes the subscript R is missed off the notation for equivalence classes if the relation in question is clear from the context; i.e. you could write $[a]$ instead of $[a]_R$.

If we were to take the set of natural numbers $\mathbb{N}$ and define the R so that aRb if a and b are both even or both odd. This would mean that $\mathbb{N}$ would be split into two equivalence classes, that would form a partition, for example, $\mathbb{N} = [2] \cup [7]$.

Mathematically, these are the principles of equivalence relations and equivalence classes:

- $a \in [a]$
- $b, c \in [a]_R \implies bRc$
- For any a, b either $[a] = [b]$ or $[a] \cap [b] = \emptyset$

3.3 Operations on Relations

Just like sets, you can perform operations such as union and intersection on relations. They are easily defined in a mathematical sense:

Union

In order to create a new relation Z that is the union of the two relations X and Y, you would do:

$$Z = X \cup Y = \{(a, b) \in Z | (a, b) \in X \vee (a, b) \in Y\}$$

An example might be if you take the union of the '*is father of*' relation and the '*is mother of*' relation, then you'll get an '*is parent of*' relation.

Intersection

Similar to the union, except you use an `and` operator instead of an `or`:

$$Z = X \cap Y = \{(a, b) \in Z | (a, b) \in X \wedge (a, b) \in Y\}$$

An example might be if you take the intersection of the relation '*person resides in Ali G*' and '*doesn't go to Manchester University*', you'd get a relation describing lots of rather suspicious looking Man Met students.

Complement

To find the complement of a relation, you just define another relation based on the pairs not defined in the first one:

$$Z = X^c = \{(a, b) \in Z | (a, b) \notin X\}$$

A complement operation would have the effect of transforming the relation '*are married*' to '*not married*'.

Inverse

The inverse of a relation is where the direction of all the arrows on the digraph are reversed:

$$Z = X^{-1} = \{(a, b) \in Z | (b, a) \in X\}$$

Inverting the relation '*is older than*' would yield an '*is younger than*' relation.

3.4 Relational composition

If one relation (R_1) is defined from A to B and another (R_2) from B to C, then it is possible to define a third relation that is composed of the first two that is defined from A to C. This is relational composition:

$$R_2 \circ R_1 = \{(a, c) \in A \times C | \exists b \in B, (a, b) \in R_1 \wedge (b, c) \in R_2\}$$

Note the order of the operands here

3.5 P Closures

In terms of relations, a P Closure is constructed when you add the minimum number of pairs to a relation in order to give the relation the specific property P. For example, the transitive closure of a relation R is a new relation based on R, but with all the extra pairs (which could be no extras) required to make the relation transitive.

If P is reflexive, symmetric or transitive, then a closure can *always* be constructed.

3.6 Ordered sets

A *partial order* is a transitive, anti-symmetric and reflexive relation. A partially ordered set (aka *poset*) is a set together with a partial order. One example of this the the set of integers $\mathbb{Z}$, which when taken together with the operator '$\geq$' becomes a partially ordered set, since any integer in the set can be compared to any other integer in the set.

A partial order that can compare *any element* is referred to as a total order.

3.6.1 Maximal and minimal elements

A minimal element m of a poset X with a partial ordering $\prec$ is one where:

$$\forall a \in X, m \prec a$$

A maximal element is similar:

$$\forall a \in X, a \prec m$$