UNIVERSITY OF BIRMINGHAM

School of Computer Science

Autumn Semester 2021

06-35324

Mathematical Foundations of Computer Science

Course notes

Contents

1 The natural numbers

1.1 The laws of arithmetic

For us, the **natural numbers** begin with zero:

$$0, 1, 2, 3, 4, \dots$$

It is useful to have a symbol for the collection of *all* natural numbers; traditionally, one uses $\mathbb{N}$. Instead of "collection", we also say "set". So $\mathbb{N}$ denotes the set of all natural numbers, starting with zero. Note that this set has *infinitely* many elements.

On the natural numbers we have the basic arithmetic operations of addition and multiplication. The symbol for addition is written "$+$", the one for multiplication is "$\times$". Some texts use a simple dot to denote multiplication, as in $3 \cdot 4 = 12$. Also, the multiplication symbol is not always written, as in $2x = x + x$.

The two arithmetic operations satisfy certain laws, which are familiar to you from your early school days:

1: The laws of arithmetic

$$\begin{array}{rclrcll}
a+0 & = & a & a\times 1 & = & a & \text{(neutral elements)}\\
a+b & = & b+a & a\times b & = & b\times a & \text{(commutativity)}\\
(a+b)+c & = & a+(b+c) & (a\times b)\times c & = & a\times(b\times c) & \text{(associativity)}\\
\multicolumn{6}{c}{a\times(b+c) = a\times b + a\times c} & \text{(distributivity)}\\
\multicolumn{6}{c}{a\times 0 = 0} & \text{(annihilation)}
\end{array}$$

More laws can be *derived* from the ones listed. Here is an example:

2

You can call this a *proof* of the law $(x+y)\times z = x\times z + y\times z$. Many more useful laws can be derived in this way from the ones given in Box 1. You can try to derive some yourself in the style of Box 2; the exercises at the end of this chapter contain some suggestions.

1.2 Beyond equations

The arithmetic operations on the natural numbers have properties that we can not express as a simple universally valid equation. To give an example, here are two important laws:

3: The cancellation laws

If $a + c = b + c$ is true, then $a = b$ is also true.
If $a \times c = b \times c$ is true, and $c \neq 0$, then $a = b$ is also true.

I am sure these rules are as familiar to you as the ones in Box 1 but you would not be able to derive them from those. In fact, soon we will see a situation where the laws of Box 1 hold but cancellation does not.

Cancellation can be used to derive the annihilation law from the others:

4

Here is a statement that is true about the natural numbers, which is even more involved than the cancellation laws:

5: Division with remainder

> Given a number $b \neq 0$, every natural number a can (uniquely) be written in the form $m \times b + r$ where m and r are natural numbers and $r < b$ holds.

Is that it, then? Not at all. But you may begin to wonder whether it is possible to write down laws *from which all other laws that hold for the natural numbers can be derived*. This question was tackled by mathematicians towards the end of the 19th century. The most successful answer was given by Giuseppe Peano (1858–1932), and it goes back to origins of the natural numbers in *counting*. Rather than use addition and multiplication straightaway, this theory uses only the operation **successor**, which you could also call "next number", or "$a+1$", or "a++"; we will write it as "$s(a)$".

6: The Peano Axioms, part 1

1. 0 is a natural number.
2. If a is a natural number then so is $s(a)$.
3. A number of the form $s(a)$ is always different from 0.
4. If $s(a)$ and $s(b)$ are equal, then a and b are equal.

These look harmless and they are. It is the next (and final) one that makes Peano's approach so powerful:

7: Peano's Axiom of Induction

5. If $P(x)$ is a property of natural numbers that

(ground case)	holds of 0, and	[so $P(0)$ is true]
(inductive step)	holds of $s(x)$ whenever it holds of x,	[so $P(x)$ implies $P(s(x))$]

then P holds of all the natural numbers.

Peano then defined addition by reducing it to counting. Here are his two clauses for addition:

$$\begin{aligned} a+0 &= a \\ a+s(b) &= s(a+b) \end{aligned}$$

Let's use these two equations plus his axioms to prove that his addition is associative, $(a+b)+c = a+(b+c)$:

8

We ask again, are the Peano axioms sufficient to prove every valid property of the natural numbers? The (negative) answer was given by Kurt Gödel (1906–1978), 50 years after Peano published his axioms.

1.3 Place value systems

According to Peano, every natural number can be written as $s(s(s(s(\ldots(s(0))\ldots))))$ but this would take a lot of space. Instead we use the **place value system**, gifted to us by the Arabs: Given a **base** $b > 1$ and **digits** $0, 1, \ldots, b-1$, we can write every natural number as a string of digits. The value of such a string $d_n d_{n-1} \ldots d_0$ is given by adding multiples of powers of b:

$$d_n \times b^n + d_{n-1} \times b^{n-1} + \ldots + d_1 \times b^1 + d_0 \times b^0 \quad \text{or} \quad d_n b^n + d_{n-1} b^{n-1} + \ldots + d_1 b + d_0 \text{ for short.}$$

This notation is highly economical because short strings can be used to denote very large numbers. (As an aside, the number of digits required to write out the number a in such a system is "very close" to the **logarithm** of a with regards to base b.) Furthermore, by memorising the sums/products of individual digits we can add/multiply numbers of arbitrary size very efficiently. You spent a good part of your primary school years doing just that.

If we write the expression $d_n \times b^n + d_{n-1} \times b^{n-1} + \ldots + d_1 \times b + d_0$ for the number a slightly differently (using the distributivity law)

$$(d_n \times b^{n-1} + d_{n-1} \times b^{n-2} + \ldots + d_2 \times b + d_1) \times b + d_0$$

then we see that the lowest-value digit d_0 is obtained from a as the *remainder* from dividing a by b, as expressed in Box 5 above. We can then continue the process with the quotient $d_n \times b^{n-1} + d_{n-1} \times b^{n-2} + \ldots + d_2 \times b + d_1$ etc and get all the digits of a for the base b representation. As an example let us represent 59 in base $b = 7$:

9

and we get the representation 113 in base 7, sometimes written as 113_7.

1.4 Natural number representation in a computer

A computer knows nothing about numbers (or anything else for that matter). How do we use its hardware to represent the mathematical concept of a natural number? A common unit in a cpu is a register of 32 bits, where each bit can be in one of two states, traditionally called "0" and "1". Thus a register can be in any one of 2^{32} different states, depending on the state of each of its 32 bits. It is now up to us to interpret these "bit patterns" as natural numbers. The usual way for doing so is to view the bit pattern as the digit representation of a number in base 2. An example:

10

This works very well and there is circuitry (in the ALU, or arithmetic logical unit) to implement addition, multiplication, and a number of other elementary operations. However, we have a big problem arising from the fact that registers are of a fixed size, that is, there are only 32 bits to represent binary digits. What happens when the result of an operation requires more than 32 digits? The answer has two parts:

1. The excess digits are available for only a short moment in the cpu.
2. Java ignores them.

The consequence of this is that the answer you get from an arithmetic operation on Java `int` variables is only correct up to multiples of 2^{32}. Because of this, the numbers available in a cpu are better imagined as sitting on a circle rather than on the number line you are familiar with from school:

11

Nevertheless, and this is extremely important, the arithmetic laws listed in Box 1 are still valid for the 2^{32} numbers we do have. This means that the algebraic manipulations you learned in school can still be applied when writing computer programs, and will not change the outcome of any calculations. However, the fact remains that the answer from a calculation can be off by multiples of 2^{32}.

Finally, Java does not actually support this "sign-less" interpretation of bit patterns, but its "ancestor" C does. When you define a variable as `unsigned int` in C, then you get a block of memory consisting of four bytes (so 32 bits) the contents of which are interpreted exactly as described above.

Exercises

1. Derive the equation $(x+y)^2 = x^2 + 2xy + y^2$ from the basic laws of arithmetic in the style of Box 2. (Recall that x^2 is just shorthand for $x \times x$.)
2. Prove by induction that the sum $2^0 + 2^1 + 2^2 + \cdots + 2^a$ equals $2^{a+1} - 1$.
3. Write out the number 2020 in base 3.
4. What is the largest number that can be represented in a 32-bit `unsigned int` variable?
5. Which of the Peano axioms is *not* valid for 32-bit unsigned integers?

Practical advice

In the exam, I expect you to be able to

- derive an equation from the basic laws of arithmetic in the style of Box 2;
- prove a statement about natural numbers by induction;
- represent a natural number in an arbitrary base.

I will also expect you

- to know the notation $\mathbb{N}$ for the set of natural numbers;
- to understand the consequences of computer arithmetic being limited to numbers that can be represented as 32 binary digits.

However, I will **not** expect you to memorise the various laws discussed in this chapter; if needed, they will be given on the exam question paper.

More information

Most textbooks on *Discrete Mathematics* start with a discussion of the natural numbers. You will find a wide choice of such books in the University Library. If there is one that you like particularly well, then you may want to consider purchasing a copy for yourself.

At the end of Part 1.2 I mentioned Gödel's result about the impossibility to axiomatise the natural numbers, his "First Incompleteness Theorem". You could have a look at `https://en.wikipedia.org/wiki/G%C3%B6del's_incompleteness_theorems` and `https://plato.stanford.edu/entries/goedel-incompleteness/` to get a glimpse of what this is all about, and why Gödel's work became so famous.