Math Week 1

Lecture 1 The natural numbers

The pitfalls of computer arithmetic

- 10^{10} = 1,1410,065,408 (in C program)

1.1 The laws of arithmetic

- neutral elements $a + 0 = a$ $a \times 1 = a$
- commutativity $a + b = b + a$ $a \times b = b \times a$
- associativity $(a + b) + c = a + (b + c)$ $(a \times b) \times c = a \times (b \times c)$
- distributivity $a \times (b + c) = a \times b + a \times c$
- annihilation $a \times 0 = 0$

These actually hold for computer integers as well

1.2 Beyond equations

- additive cancellation $a + c = b + c \Longrightarrow a = b$
- multiplicative cancellation $c \ 6 = 0 \ \& \ a \times c = b \times c \Longrightarrow a = b$

Additive cancellation also holds for computer integers
But multiplicative cancellation does not hold for computer integers

- Peano's Axioms

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
5. If *P(x)* is a property of natural numbers that (ground case) holds of 0, and (inductive step) holds of *s(x)* whenever it holds of *x* then *P* holds of all the natural numbers

The Axiom of Induction can be used to prove that some properties P(a) and true for all natural numbers a

1.3 Place value systems

- **Place value system**
  - **base** *b*>0 and **digits** 0,1,....,*b*-1
    $d_n \times b^n + d_{n-1} \times b^{n-1} + ... + d_1 \times b^1 + d_0 \times b^0$
    or
    $d_n b^n + d_{n-1} b^{n-1} + ... + d_1 b + d_0$

1.4 Natural number representation in a computer

- The bit pattern as the digit representation of a number in base 2
- What happens when the result of an operation requires more than 32 digits(There are only 32 bits to represent binary digits)?
  i. The excess digits are available for only a short moment in the cpu
  ii. Java ignores them

Left associative: Read from left to right
Right associative: Read from right to left

Ambiguous: without either Precedence or Associativity

Lecture 2 The integers

2.1 The arithmetic laws of integers

- We only need to assume that for every integer a there is another integer denoted by -a for which a + (-a) = 0 holds
- From this we can prove (additive) cancellation
- From this we can prove annihilation

- From this we can prove double negation: -(-a) = a
- From this we can prove minus times minus equals plus:
  (-a) x (-b) = a x b

2.2 Rings

```
The laws of rings
      a + 0 = a                   a x 1 = a            (neutral elements)
      a + b = b + a               a x b = b x a        (commutativity)
   a + (-a) = 0                                        (additive inverse)
(a + b) + c = a + (b + c)   (a x b) x c = a x (b x c)  (associativity)
                 a x (b + c) = a x b + a x c           (distributivity)
```

2.3 Integers in computers

- Java's int variables are based on 32-bit registers
- All calculations are done modulo 2^{32}
- The bit patterns from 100...000 to 111...111 are interpreted as **negative numbers**

2.4 Modulo arithmetic

- Computng "modulo m" can be done for any m > 1. We get the ring Z_m which has exactly m different elements
- Calculations in Z_m can be thought of in two different ways:
  i. We can take the numbers from 0 to m - 1 as the standard members of Z_m, perform calculations with them as we would in Z, then reduce the result to an answer between 0 and m - 1 at the end. Example in Z_7
     3 x 5 = 15 in Z
     = 1 modulo 7
     so in Z_7 we have 3 x 5 = 1
  ii. Alternatively, we can do all calculations in Z and use = for comparisons, instead of =.
- Computer integers implement calculations in $Z_2$32 and adopt the first approach internally, but when reporting the result back to the user, the numbers between 2^{31} and 2^{32} - 1 are converted to negative numbers by subtracting 2^{32}