COMP11120 Notes

Todd Davies, Chris Williamson

October 30, 2013

Contents

1 Discrete Structures

1.1 Terminology

A *structure* consists of certain *sets*. It also contains *elements* of these sets, *operations* on these sets and *relations* on these sets.

1.2 Number systems to learn

The following number must be learnt:

$\mathbb{N}$	The set of natural numbers (all whole numbers from 0 to ∞)
$\mathbb{Z}$	The set of integers (all whole numbers from $-\infty$ to ∞)
$\mathbb{Q}$	The set of rational numbers (any integer divided by any other integer e.g. $\frac{5}{4} = 1.25$)
$\mathbb{R}$	The set of real numbers (all finite and infinite decimal numbers)

1.2.1 Operations

Each number system has a set of valid operations that can be performed on elements in that system. Number systems only contain operations that will produce an output that is still within the number system.

For example, the number system $\mathbb{N}$ contains the operations of addition and multiplication. This is because the summation of any two positive integers will *always* be a member of $\mathbb{N}$, and the same goes for multiplication.

However, you may be wondering why subtraction and division aren't included in this number system. This is because for some numbers, the result of subtraction or division won't be inside the set $\mathbb{N}$. An example of this would be subtracting 4 from 2. Even though both of the operands are inside $\mathbb{N}$, the answer isn't.

Different sets may have different operations available. For example, you can concentrate any two members of the set $\mathbb{S}tring$ and end up with another $\mathbb{S}tring$.

Types of operation Operations that have two operands either side of them are called *infix* operations. An example of an infix operation is addition. Infix operations are also referred to as *binary* operations since they have *two* operands.

Commutativity If an operation is commutative, the order of the operands doesn't matter. For example, addition is commutative since:

$$a + b = b + a$$

Subtraction however, isn't commutative:

$$a - b \neq b - a$$

Associativity An operation is associative if inserting or changing brackets doesn't change the outcome of the operation. For example, multiplication is associative since:

$$(a \times b) \times c = a \times (b \times c)$$

An operation may only be commutative or associative if it is commutative or associative for *all* elements of the set that the operation supports.

Distinguished elements A set may contain distinguished elements that have strange effects on certain operations in the set. An example is the number 1. If we multiply *something* by 1, then the result will always be the same *something*. The same goes for 0 with addition. Because of this, we refer to 1 and 0 as distinguished elements of the set $\mathbb{Z}$.

1.2.2 Relations

Each of the sets $\mathbb{N}, \mathbb{Q}, \mathbb{Q}, \mathbb{R}$ carries binary comparison relations $\leq$ and $<$. Different sets (such as $\mathbb{S}tring$) may have other relations, such as:

- Is a section of
- Is an initial section of
- Occurs in

All relations return values in the set $\mathbb{B}ool$.

1.3 Bases

Conventionally, we count using base 10. Base 10 includes, you guessed it, ten different symbols from 0 through to 9.

Sometimes however, it is convenient to count using different bases. Popular bases include:

Base n	**Member symbols**	**Name**
$n = 2$	$\mathbb{Z}_2 = \{0, 1\}$	Binary
$n = 8$	$\mathbb{Z}_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$	Octal
$n = 10$	$\mathbb{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7\}$	Decimal/Denary
$n = 16$	$\mathbb{Z}_{16} = \{0, 1, 2, \ldots, 9, A, B, C, D, E, F\}$	Hexadecimal

1.3.1 How to read numbers in any given base

The formula for reading a number in a given base is as follows:

$$\sum_{i=0}^{k} a_i b^i$$

Where the number you're trying to read takes the form $a_k, a_{k-1}, \ldots, a_2, a_1, a_0$ and b is the base you're using.

Example 1 Lets apply the formula to the base 10 number 27385:

$$\begin{aligned}
27385 &= (5 \times 10^0) + (8 \times 10^1) + (3 \times 10^2) + (7 \times 10^3) + (2 \times 10^4) \\
&= (5 \times 1) + (8 \times 10) + (3 \times 100) + (7 \times 1000) + (2 \times 10000) \\
&= 5 + 80 + 300 + 7000 + 20000 \\
&= 27385
\end{aligned}$$

Example 2 Lets apply the formula to the base 16 number $F00BA4$:

$$\begin{aligned}
F00BA4 &= (4 \times 16^0) + (A \times 16^1) + (B \times 16^2) + (0 \times 16^3) + (0 \times 16^4) + (F \times 16^5) \\
&= (4 \times 16^0) + (10 \times 16) + (11 \times 256) + (0 \times 4096) + (0 \times 65536) + (15 \times 1048576) \\
&= 4 + 160 + 2816 + 0 + 0 + 15728640 \\
&= 15731620
\end{aligned}$$

1.3.2 Changing from base 10 to base n

In order to change into base n from base 10, we just repeatedly divide by n and use the remainder as the value for base n. Here are a few examples:

Example 1 Convert 893 into base 2.

$$\begin{array}{rcll}
893 \div 2 & = & 446 & \text{r1} \\
446 \div 2 & = & 223 & \text{r0} \\
223 \div 2 & = & 111 & \text{r1} \\
111 \div 2 & = & 55 & \text{r1} \\
55 \div 2 & = & 27 & \text{r1} \\
27 \div 2 & = & 13 & \text{r1} \\
13 \div 2 & = & 6 & \text{r1} \\
6 \div 2 & = & 3 & \text{r0} \\
3 \div 2 & = & 1 & \text{r1} \\
1 \div 2 & = & 0 & \text{r1}
\end{array}$$

Reading up from the bottom, we can see that the binary (base 2) representation is 1101111101.

Example 2 Convert 893 into base 9.

$$\begin{array}{rcll}
893 \div 9 & = & 99 & \text{r2} \\
99 \div 9 & = & 11 & \text{r0} \\
11 \div 9 & = & 1 & \text{r2} \\
1 \div 9 & = & 0 & \text{r1}
\end{array}$$

Reading up from the bottom, we can see that the nonal (base 9) representation is 1202.

Example 2 Convert 893 into base 16.

$$\begin{array}{rcll}
893 \div 16 & = & 55 & \text{r13} \\
55 \div 16 & = & 3 & \text{r7} \\
3 \div 16 & = & 0 & \text{r3}
\end{array}$$

Reading up from the bottom, we can see that the hexadecimal (base 16) representation is $3, 7, 13$ or $37D$.

1.4 A structure for the integers

1.4.1 The basic data

The set $\mathbb{Z}$ of all integers:

$\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$

includes the subset $\mathbb{N}$ of all natural numbers together with the negative integers. We will be using three basic binary operations on the carrier set $\mathbb{Z}$: addition, multiplication and subtraction.

The standard notation for there is +, x and - used as infix operations.

It is conventional to write xy as an abbreviation for $x\text{x}y$

Sometimes different notations are used, for example, in programming * is usually used.

These operations illustrate two properties which an arbitrary binary operation may or may not have.

Both + and x are **commutative** but - is not.

1. For all integers *x,y* both, $x + y = y + x$ and $xy = yx$
2. There are integers *x,y* with $x - y \neq y - x$.

Commutativity ensures that for most purposes the order in which the two arguments are consumed is irrelevant.

Both + and x are **associative** but - is not.

1. For all integers *x,y,z* both $(x + y) + z = x + (y + z)$, $(xy)z = x(yz)$.
2. There are integers *x,y,z* with $(x - y) - z \neq x - (y - z)$.

Associativity ensures that for most purposes brackets are not needed to punctuate an expression. For instance, we can make sense of $x + y + z$ and xyz since it doesn't matter where the brackets are, the resulting values are the same.

Below are the definitions of commutativity and associativity.

N.b. Commutativity and associativity don't always have to go together. An example is the average of two numbers, what is that?

1. $\circledast$ is **commutative** if and only if

$$a1 \circledast a2 = a2 \circledast a1$$

for all $a1, a2, \in, A$

2. $\circledast$ is **associative** if and only if

$$(a1 \circledast a2) \circledast a3 = a1 \circledast (a2 \circledast a3)$$

for all $a1, a2, a3, \in, A$

Some numbers have special effects on operations in a set. Some examples are 0 and 1. When 0 is added to a number, the result is still that number, and when multiplying a number by 1, the result is still that number. This means for the set $\mathbb{Z}$, 0 and 1 are neutral elements for the operations addition and multiplication respectively.

1.5 The formal language

We use the formal language to talk about $\mathbb{Z}$. Expressions in the formal language are built up from atoms in a recursive fashion, so an expression may be $(x \circ y)$ where x and y are expressions and $\circ$ is one of the three operations $(+, -, \times)$. The brackets are important in the formal language, since they ensure that strings can only be parsed in one way.

Strings that contain literals that aren't neutral elements or have invalid bracketing are not valid in the formal language. Examples may include:

$$(x + y + z)$$
$$(x + 2)$$
$$x((y$$

We can use a parse tree to show how an expression is built up from it's identifiers $(a, b, c \ldots)$, constants $(0, 1 \ldots)$ and operations $(+, -, \times \ldots)$.

For example, the parse tree of $(x \times (y + z))$ is:

$$\dfrac{x \quad \dfrac{y \quad z}{(y + z)}\,(+)}{(x \times (y + z))}\,(\times)$$

Often, especially with large parse trees, it's a pain to write so many identifiers. Because of this, it is a convention to replace identifiers with dots after they've been used once, like so:

$$\dfrac{x \quad \dfrac{y \quad z}{\cdot}\,(+)}{\cdot}\,(\times)$$