Lecture 00.04 Numeral systems

The following is a myth in the good sense of the term.

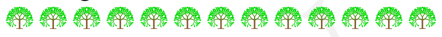
00.04.1 The myth of Niles and Pepper

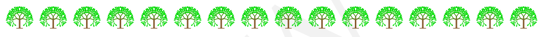
Niles was an unusual boy with great hair, living in a time before number systems. Quantities were familiar, but symbolic representations of them were not. Niles lived in a grove of trees. A grove on the other side of the hill was home to his friend Pepper, a sassy, no-nonsense girl.

One day Niles and Pepper were walking together and, as children do, began arguing about whose grove had more trees. If the argument had been about who had more skipping stones, they could have simply matched up stone-for-stone to discover who had more. But this was impractical with the trees. Niles had an insight:

> *We can represent each tree by a drawing and match these to determine who has more.*

It went something like this.

Niles: 

Pepper: 

Not to be discouraged, Niles proposed they compare, instead, the number of trees on each's entire side of the hill. However, there were many more trees, so Pepper suggested they simply draw the symbol ⊤ to represent each tree, to save time. The results were no more-satisfying to Niles.

Niles: ⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤

Pepper: ⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤⊤

Niles pushed on: let's include the neighboring hill on each side. With so many more trees, Pepper suggested a shorthand notation.

> *We can compactly represent the number of trees with two symbols* ◯ *and* | *used in combination.*

She explained it to Niles by counting up:

- ␣ = ◯
- ⊤ = |
- ⊤⊤ = |◯
- ⊤⊤⊤ = ||
- ⊤⊤⊤⊤ = |◯◯
- ⊤⊤⊤⊤⊤ = |◯|

- $\top\top\top\top\top\top = ||\bigcirc$
- $\top\top\top\top\top\top\top = |||$.

That is, each position could take on two symbols, but the *position* of each symbol denoted its "weight." The far-right symbol represented either a lack $\bigcirc$ or presence $|$ of a single tree. The next symbol to the left was the "overflow" from the first symbol, and therefore represented the number of *pairs of trees*. The next symbol to the left was the next overflow, representing *pairs of pairs of trees*.

What fun! They started counting and Pepper immediately recognized a process improvement:

> *If we use more symbols, we can represent numbers even more-compactly.*

Pepper suggested a symbol for each digit of their hands:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9. \tag{00.1}$$

Now they could count on their fingers:

- $\textvisiblespace = 0$
- $\top = 1$
- $\top\top = 2$
- $\top\top\top = 3$
- $\top\top\top\top = 4$
- $\top\top\top\top\top = 5$
- $\top\top\top\top\top\top = 6$
- $\top\top\top\top\top\top\top = 7$
- $\top\top\top\top\top\top\top\top = 8$
- $\top\top\top\top\top\top\top\top\top = 9$
- $\top\top\top\top\top\top\top\top\top\top = 10$.

That is, the second symbol now represented a group of ten of the group to the right: the rightmost, the number of trees; to its left, the number of tens of trees; to its left, the number of tens of tens (hundreds); to its left, the number of tens of hundreds (thousands); etc.

Pepper still had more trees.

00.04.2 Positional numeral systems

numeral system

Niles and Pepper, when they represented a tree by $\top$, created a very simple *numeral system*: a way of representing quantities with symbols called *numerals*. Once they recognized the value of including multiple symbols (◯ and |, at first) and endowed the *position* of each numeral with significance, the system became a *positional numeral system*. The number of numerals used is called the system's *base*: two for the system with ◯ and | and ten for that with 0–9. Base-2 systems are called *binary*, and typically use the symbols 0 and 1 instead of ◯ and |. *Base-10* systems are those with ten numerals, the most common of which is called the *Hindu-Arabic numeral system* and uses the *Arabic numerals* 0-9.

numeral

positional numeral system

base

binary

base-10

Hindu-Arabic numeral system

Arabic numerals

Let's consider the meaning of the Arabic number 937. It means 9 *hundreds*, 3 *tens*, and 7 *ones*. A corresponding arithmetic representation is

$$9 \times 10^2 + 3 \times 10^1 + 7 \times 10^0.$$

Similarly, the binary number 1011 has the meaning 1 *pair of pairs of pairs of ones*, 0 *pair of pairs of ones*, 1 *pair of ones*, 1 *one*. Let's convert this binary representation to base-10:

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11.$$

This highlights an important ambiguity: how can we tell in which numeral system 11 is written? We cannot, so we must either rely on context, explicitly state, or add subscripts, as in

$$1011_2 = 11_{10}. \tag{00.2}$$

As a convention, we restrict interpretations of numeral system-denoting subscripts to base-10.

Now we introduce nuanced versions of the above numeral systems.

00.04.3 Decimal numeral system

Representing non-integer numbers is done with a *radix point*, often ".". The *decimal numeral system* is the Hindu-Arabic system extended to include non-integer numbers. Digits (decimal numerals) increasingly right of the radix point (called a *decimal point* in the decimal system) represent *tenths*, *hundredths*, *thousands*, etc. For instance, the decimal 2.73 would be

radix point

decimal numeral system

decimal point

$$2 \times 10^0 + 7 \times 10^{-1} + 3 \times 10^{-2}.$$

00.04.4 Hexadecimal numeral system

The *hexadecimal numeral system* extends the decimal system with an addi-

hexadecimal numeral system

tional six numerals, borrowed from the beginning of the Latin alphabet, to have a total of sixteen numerals:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}.$$

As we will see, this base-16 system provides a convenient way to represent the contents of computer memory.

It is conventional to begin hexadecimal numbers with the prefix "0x" as in 0x9A78 0D38.

00.04.5 Converting to and from decimal

Converting from a base-b number x_b with digits $x_n x_{n-1} \cdots x_0.x_{-1}x_{-2} \cdots x_{-m}$ to decimal is straightforward. Represent each numeral in base-10, then use the formula

$$x_{10} = \sum_{i=-m}^{n} x_i b^i. \tag{00.3}$$

For instance, if $x_2 = 1010.01$,

$$\begin{aligned} x_{10} &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= 10.25. \end{aligned}$$

Similarly, if $x_{16} = \mathrm{B8.F}$,

$$\begin{aligned} x_{10} &= 11 \times 16^1 + 8 \times 16^0 + 15 \times 16^{-1} \\ &= 184.9375. \end{aligned}$$

Converting from decimal into a base-b numeral system can be accomplished by the following procedure.

- For the integer part of the number, successively divide by the base b_{10}, represented in base-10. The remainder x_b, represented in base-b, of each step is the base-b numeral in that position, from right-to-left.
- For the decimal part of the number, successively multiplying by the base b_{10}. The overflow x_{-b} of 1_{10} and above, at each step, is the corresponding base-b numeral in that position, from left-to-right.

Note that division and multiplication in the conversion process are the usual base-10 versions. Technically, this process can be used for converting between other numeral systems, but it is not recommended due to our unfamiliarity with division and multiplication in these numeral systems.

Example 00.04-1 decimal to binary and hex

1. Convert 14_{10} to binary.
2. Convert 14_{10} to hexadecimal.
3. Convert 421.73_{10} to binary.

---

1. The following table shows the division.

b/divisor	dividend/quotient	remainder
	14	
2	7	0
2	3	1
2	1	1
2	0	1

   Therefore, $14_{10} = 1110_2$.
2. There is no need to divide because $14_{10} \leqslant 15_{10}$, the number of hex numerals. Therefore, $14_{10} = E_{16}$.
3. For the integer part, the following table shows the division.

b/divisor	dividend/quotient	remainder
	421	
2	210	1
2	105	0
2	52	1
2	26	0
2	13	0
2	6	1
2	3	0
2	1	1
2	0	1

   Therefore, $421_{10} = 110100101_2$. Now for the number right of the decimal point.

b/factor	factor/product	overflow
	0.73	
2	.46	1
2	.92	0
2	.84	1
2	.68	1
2	.36	1
2	.72	0
2	.44	1
2	.88	0
2	.76	1
2	.52	1
2	.04	1
2	.08	0
2	.16	0
2	.32	0
2	.64	0
2	.28	1
2	.56	0
2	.12	1
2	.24	0
2	.48	0
2	.96	0
2	.92	

This last row is identical to the second row. Therefore, an infinite loop will occur and $421.73_{10} = 110100101.1\overline{01110101110000101000}_2$. This has profound implications: an exact decimal value of 421.73 must be rounded to be stored as binary. This introduces rounding error even when it might appear we know the number, exactly. (Note that this is for *floating point* representations of the decimal number, which is common. Alternatively, five integers could represent the decimal, exactly.) See 01.01.1.3 for exactly this (BCD).

00.04.6 Converting between hex and binary

Binary numbers can be easily converted hexadecimal and vice-versa. In Lecture 01.01 these conversions will be motivated. There are $2^4 = 16$ unique four-numeral binary numbers and 16 hex characters (which is no coincidence). This allows us to write each grouping of four binary numerals, called a *nibble*, as a single hex character. It is often easiest to convert each nibble to base-10, then (trivially) to hex. For instance, 1101_2 is

nibble

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}.$$

The thirteenth hex numeral is D.

Similarly, one can convert a hex numeral to a nibble by converting it first to decimal, then to binary.

00.04.7 Signed binary numeral system

The signed binary numeral system, often called the *two's complement* numeral system, is used to represent both positive and negative numbers in binary form. When encountering a signed binary number, first consider the leftmost numeral: if it is 0, the number it represents is positive or zero and the usual binary-to-decimal conversion holds; if it is 1, the number it represents is negative and must undergo the two's complement *operation* before the usual binary-to-decimal conversion holds for its negation.

two's complement

The two's complement operation can be performed by flipping all the bits ($0 \to 1$ and $1 \to 0$) and adding 1. For instance,

$$1101\,0110 \xrightarrow{\text{flip bits}} 0010\,1001 \xrightarrow{\text{add one}} 0010\,1010.$$

This result can be converted to decimal in the usual way: $0010\,1010_2 = 42_{10}$. Therefore $1101\,0110_2 = -42_{10}$.

Of course, this means that an n-numeral in two's complement binary stores not (as would unsigned binary)

$$0, 1, \cdots 2^n - 1$$

but

$$-2^{n-1} + 1, -2^{n-1} + 2, \cdots - 1, 0, 1 \cdots 2^{n-1}.$$

In both unsigned and (two's complement) signed binary, however, a total of 2^n numbers are represented by n binary numerals.