Digital Systems and Information Representation

CSE 361S

What is Binary?

- Underlying base signals are two-valued:
  - 0 or 1
  - true or false (T or F)
  - high or low (H or L)
- One “bit” is the smallest unambiguous unit of information
- Propositional calculus helps us manipulate (operate on) these base signals

Operations in Propositional Calculus

AND \( a \cdot b = c \)
\[ a \quad b \rightarrow c \]
c is true if \( a \) is true and \( b \) is true

OR \( a + b = c \)
\[ a \quad b \\rightarrow c \]
c is true if \( a \) is true or \( b \) is true

NOT \( a' = b \)
\[ a \\rightarrow b \]
b is true if \( a \) is false

An Example

- \( a \) passed microeconomics course
- \( b \) passed macroeconomics course
- \( c \) passed economics survey course
- \( d \) met economics requirement

\[ d = a \cdot b + c \]
\[ a \quad b \rightarrow c \rightarrow d \]

Boolean Algebra

- Boolean algebra (named after 19th century mathematician George Boole) lets us manipulate and reason about expressions of propositional calculus
- Systems based on this algebraic theory are called “digital logic systems”
- All modern computer systems fall in this category

Physical Representation

- Positive logic convention
  - Binary value (1 or 0) is represented by the voltage on a wire (H or L)
  - true, 1 voltage greater than threshold \( V_H \)
  - false, 0 voltage less than threshold \( V_L \)
  - Voltage gap between \( V_H \) and \( V_L \) provides safety margin to limit errors
That’s Not Enough!

• We are interested in representing signals that have more than just two values
  – numbers
  – text
  – images
  – audio
  – video
  – and much more
• Let’s look at a sequence of number systems (unveiling some history in the process)

Start with “counting” numbers

• 1, 2, 3, ... (positive integers)
• good at measuring how much stuff I have
  – e.g., 7 cows and 2 goats
• closed under addition and multiplication
  – if a,b are numbers, a + b is too
  – if a,b are numbers, a × b is too

What if I don’t have any stuff?

• incorporate 0 into number system
  – I have 0 pigs
• still closed under addition and multiplication
• how do we represent zero in Roman numerals?
  – we don’t!
  – the Romans hadn’t yet invented zero

What if I take away more than I have?

• new concept: negative numbers
  – I owe my neighbor 3 chickens, vs.
  – I own -3 chickens
• closed under subtraction as well
• can solve equation x + a = 0 for any integer a

Why can’t I divide my stuff 3 ways?

• incorporate rational numbers
  – I have 2 pigs, so when I die...
  – each of my 3 sons gets 2/3 pig
• now closed under division
• can now solve equation ax + b = 0 for arbitrary numbers a,b if a ≠ 0

How long is this line?

• incorporate irrational numbers
• defined “real” numbers
• can now solve equation x^2 – 2 = 0
• Note pattern of names:
  – positive to negative
  – rational to irrational
But what about \( x^2 + 2 = 0 \)?

- introduce complex numbers
- vector number system with 2 components:
  - \((a, b)\) \(a, b\) both “real” numbers
- obey following rules:
  - equality: \((a, b) = (c, d)\) iff \(a = c\) and \(b = d\)
  - addition: \((a, b) + (c, d) = (a + c, b + d)\)
  - multiplication: \((a, b) \times (c, d) = (ac – bd, ad + bc)\)

Interesting fact

- complex numbers with 2\(^{nd}\) component equal 0 have same properties as “real” numbers
  - \((a, 0) = (c, 0)\) iff \(a = c\)
  - \((a, 0) + (c, 0) = (a + c, 0)\)
  - \((a, 0) \times (c, 0) = (ac, 0)\)

What name to use?

- 1\(^{st}\) component of complex number is called “real”, why not follow the historical naming conventions?
  - \(\neg\) positive \(\Rightarrow\) negative
  - \(\neg\) rational \(\Rightarrow\) irrational
  - \(\neg\) real \(\Rightarrow\) imaginary
- therefore, 2\(^{nd}\) component of complex number came to be called “imaginary”

Back to our equation

- What about the equation we were trying to solve?
  - \(x^2 + 2 = 0\)
- First, rewrite as equation in complex numbers
  - \(x^2 + (2,0) = (0,0)\)
- Second, insert \(x = (0, \sqrt{2})\)
  - \((0, \sqrt{2})^2 + (2, 0) = (0, \sqrt{2}) \times (0, \sqrt{2}) + (2, 0)\)
  - \((-2, 0) + (2, 0)\)
  - \((0, 0)\)

Another interesting equation: \(x^2 + 1 = 0\)

- Initial algebraic manipulation yields:
  - \(x^2 + 1 = 0\)
  - \(x^2 = -1\)
  - \(x = \sqrt{-1}\)
- Now try with \(x = (0,1):\)
  - \((0, 1)^2 + (1, 0) = (0, 1) \times (0, 1) + (1, 0)\)
  - \((-1, 0) + (1, 0)\)
  - \((0, 0)\)
- Therefore, \(x = (0, 1) = \sqrt{-1}\)

New notation for complex numbers

- Let symbol \(i = \sqrt{-1}\)
- \((a, b)\) can now be written \(a + ib\)
- since \(a + ib = (a, 0) + (0, 1) \times (b, 0)\)
  - \(= (a, 0) + (0, b)\)
  - \(= (a, b)\)
- Note: EEs use \(j = \sqrt{-1}\), the rest of the known universe uses \(i = \sqrt{-1}\). Why is that?
How powerful are complex numbers?
• We now have a number system rich enough to solve arbitrary constant coefficient polynomial equations:
  \[ a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n = 0 \]
• If a’s complex-valued, \( n \geq 1 \), and \( a_0 \neq 0 \), there are precisely \( n \) roots in complex number system
  “Fundamental Theorem of Algebra”

How do we represent these numbers?
• A positional number system lets us represent integers. E.g., in base 10:
  \[ xyz_{10} = x \cdot 10^2 + y \cdot 10^1 + z \cdot 10^0 \]
  \[ = x \cdot 100 + y \cdot 10 + z \]
x, y, z can each have 10 possible values: 0 to 9

Base 2 (binary) works the same way
\[ xyz_2 = x \cdot 2^2 + y \cdot 2^1 + z \cdot 2^0 \]
\[ = x \cdot 4 + y \cdot 2 + z \]
x, y, z can each have 2 possible values: 0 or 1

e.g.,
- 000 = 0
- 001 = 1
- 010 = 2
- 011 = 3
- 100 = 4
- 101 = 5
- 110 = 6
- 111 = 7

Negative numbers
• With a fixed number of bits, one can represent negative numbers in a variety of ways. E.g., 4-bit binary number system:
  • unsigned range 0 to 15 (0000 to 1111)
  • unsigned integers with \( n \) bits range 0 to \( 2^n - 1 \)
  • offset or bias (e.g., -7) range -7 to 8
    subtract fixed amount (such as midpoint value)
    generally bad for computation

4-bit Sign-Magnitude
1st bit encodes sign (0 = positive, 1 = negative)
bits 2, 3, 4 magnitude ⇒ range 0 to 7 (000 to 111)
overall range -7 to +7
what about 1000? -0!

with \( n \) bits, use \( n-1 \) bits for magnitude
range \(-2^{n-1} \cdot 1\) to \(+2^{n-1} \cdot 1\)

issues:
• two representations for “0”, +0 and -0
• need significant hardware to support add, subtract

2’s (radix) complement
• Use negative weight for 1st bit:
  \[ wxyz = w \cdot -(2^3) + x \cdot 2^2 + y \cdot 2^1 + z \cdot 2^0 \]
  \[ = w \cdot -8 + x \cdot 4 + y \cdot 2 + z \]
• overall range -8 to +7
• 1st bit is still sign bit,
  with 0 = positive and 1 = negative
• only one zero: 0000
Properties of 2’s complement

• least significant n-1 bits have unaltered meaning (i.e., standard positional notation and weights apply)

• most significant bit has weight negated (instead of weight $2^{n-1}$, it is weight $-2^{n-1}$)

• range -(2^n-1) to +(2^n-1-1)

• negation operation: flip all bits, add 1, throw away carry