MIT 6.004 Computation Structures: Course Notes

Instructor: Silvina Hanlon

Spring 2017

1 Information Theory

- **Definition of Information:** Data that resolves uncertainty about a fact or circumstance.
- **Quantifying Information (Shannon, 1948):**
  - Random variable X with discrete outcomes x_i and probabilities p_i.
  - Information content of outcome x_i:

$$I(x_i) = \log_2 \frac{1}{p_i}.$$

  - Units: bits.
- **Examples:**
  - Card drawn from 52: learning suit (heart) gives $\log_2(52/13) = 2$ bits.
  - Face card (J/Q/K): $\log_2(52/12) \approx 2.115$ bits.
  - Suicide King: $\log_2(52/1) \approx 5.700$ bits.
  - Coin flip: $\log_2(2/1) = 1$ bit.
  - Two dice: $\log_2(36/1) \approx 5.17$ bits (fractional bits interpret over many trials).

2 Entropy

- **Definition:** Average information content (expected value) of random variable X.

$$H(X) = \sum_i p_i \log_2 \frac{1}{p_i}.$$

- **Interpretation:** Lower bound on the average number of bits required to encode values of X.

- **Worked Example:**
  - Outcomes $\{A, B, C, D\}$ with probabilities $\{1/3, 1/2, 1/12, 1/12\}$.
  - Information contents $\{1.585, 1, \ldots\}$, compute $H(X) \approx 1.626$ bits.

3 Encoding Schemes

3.1 Fixed-Length Encoding

- Each symbol assigned bit strings of equal length.
- Example: 4 symbols $\rightarrow$ 2 bits each.
- Supports random access.

3.2 Variable-Length Encoding

- Symbols have bit strings of differing lengths.
- Shorter codes for higher-probability symbols.
- Must ensure *uniquely decodable* (prefix-free).
- Represented via binary trees: symbols at leaves.
- Example encoding: $\{A : 00, B : 1, C : 000, D : 001\}$.

4 Huffman Coding

- Algorithm to construct optimal prefix-free variable-length code.
- Repeatedly combine two least-probable symbols/subtrees.
- Yields minimal expected code length.

5 Error Detection and Correction

5.1 Hamming Distance

- Number of differing bit positions between two codewords.
- Single-bit error changes codeword by distance 1.

5.2 Parity Check

- Add parity bit to enforce even (or odd) number of 1s.
- Detects any single-bit error (min. distance 2).

5.3 Error Correction

- To correct up to E errors, require minimum Hamming distance $2E+1$.
- Single-bit correction: distance 3 (e.g., Hamming codes).

6 Number Representations

6.1 Unsigned Binary

- N bits represent values $0 \ldots 2^N - 1$.
- Binary-to-decimal conversion: sum of bit weights.

6.2 Hexadecimal

- Radix-16 grouping of 4 bits per hex digit.
- Prefix `0x` denotes hex literals.

6.3 Signed Representations

- *Signed magnitude*: separate sign bit (inefficient, two zeros).
- *Two's complement*: high-order bit negative weight.
- Range: -2^{N-1} to $2^{N-1} - 1$; arithmetic via standard binary addition.
- Negation: bitwise complement + 1.

6.4 Worked Examples

- **Example 1: Hat of Names**
  - A hat contains 5 women and 3 men ($N = 8$ possible names).
  - You learn "the selected name is a man" $\implies$ remaining possibilities $M = 3$.
  - Information conveyed:

$$I = \log_2 \frac{N}{M} \ = \ \log_2 \frac{8}{3} \ = \ \log_2\left(\tfrac{1}{3/8}\right) \text{ bits.}$$

- **Example 2: 4-bit Two's Complement**
  - All 4-bit patterns $\implies$ $N = 16$ equally likely values.
  - You learn "the number is > 0" $\implies$ positive outcomes $M = 7$.
  - Information conveyed:

$$I = \log_2 \frac{N}{M} = \log_2 \frac{16}{7} \text{ bits.}$$

7 Two's-Complement Representation

- Uses N bits to encode signed integers in range $[-2^{N-1}, 2^{N-1}-1]$.
- **Bit weights:**

$$\overbrace{-2^{N-1}}^{\text{MSB}},\ +2^{N-2},\ 2^{N-3},\ \ldots,\ 2^1,\ 2^0.$$

- **Positive values:** MSB = 0
  - Example (6-bit): `001000`

$$= 2^3 = 8.$$

  - To extend width, prepend zeros (e.g. `00001000` for 8 in 8bits).
- **Negative values:** MSB = 1
  - Example (6-bit): `101100`

$$= -2^5 + 2^3 + 2^2 = -32 + 8 + 4 = -20.$$

  - To extend width, prepend ones (e.g. `11101100` for -20 in 8bits).
- **Negation formula:**

$$-A = \widetilde{A} + 1,$$

  where $\widetilde{A}$ is the bitwise complement.
- **Arithmetic:**

$$A - B = A + (-B).$$

  - *Example:* 6-bit $15 - 18$

$$\begin{aligned} 15 &= \texttt{001111}, \quad 18 = \texttt{010010}, \\ -18 &= \widetilde{010010} + 1 = \texttt{101101} + 1 = \texttt{101110}, \\ 15 + (-18) &= \texttt{001111} + \texttt{101110} = \texttt{111101}, \end{aligned}$$

$$\text{interpret } 111101 : \text{negate } (\texttt{000010} + 1) = \texttt{000011} = 3, \text{ so } 111101 = -3.$$

  - **Overflow:** occurs if adding two positives gives negative or two negatives gives positive.

8 Voltage–Based Image Encoding

- A black-and-white image can be represented point-by-point by a voltage: black = 0V, white = 1V, intermediate intensities = in-between voltages.

- To encode the information of Nbits per pixel, we must reliably distinguish 2^N distinct voltages in $[0, 1]$ V.
- In practice, thermal noise and instrumentation limits put a cap on N—e.g. distinguishing four levels (2bits) is easy, but a million levels (20bits) is essentially impossible.
- Continuous scan: rasterize image left-to-right, top-to-bottom → time-varying voltage waveform (early television).
- *Continuous-value processing* (COPY, INVERT) accumulates analog error ϵ at each stage; small errors blur and distort the final image.

9 Information Processing Blocks

- **COPY block:** output follows input voltage exactly (idealized).
- **INVERT block:** output $= 1 - V_{\text{in}}$ (negates intensity).
- *Composition:* wire blocks together like tinker-toys, predict system behavior by "black-box" rules without internal details.
- *Failure mode:* non-ideal gain and offset in each block → accumulated error, fidelity loss grows with system depth.

10 Digital Abstraction

- Replace continuous voltages by a two-level encoding ("0" or "1") via thresholds.
- **First cut:** single threshold V_{TH}—impractical because small noise near threshold causes bit flips.
- **Second cut:** two thresholds V_L and V_H define

$$\begin{cases} V \leq V_L & \to 0, \\ V \geq V_H & \to 1, \\ V_L < V < V_H & \text{(forbidden zone).} \end{cases}$$

- In the forbidden zone, converter may output either value or none—provides *noise margin*.

11 Combinational Digital Devices

- A device is *combinational* if it obeys the *static discipline*:
  1. Digital inputs: voltages below $V_{IL} \to 0$, above $V_{IH} \to 1$.

2. Digital outputs: voltages $\leq V_{OL}$ for "0", $\geq V_{OH}$ for "1".
3. Functional spec: for each input pattern, output is defined (truth table).
4. Timing spec: propagation delay t_{PD} bounds "valid-in→valid-out" delay.

- Composition rules:
  - No directed cycles.
  - Each input connected exactly once to an input port, constant, or one output.
  - Guarantees the assembled system is itself combinational.

12 Signaling Specifications and Noise Margins

- Separate input and output thresholds:

$$V_{OL} < V_{IL} < V_{IH} < V_{OH}.$$

- *Low noise margin:* $NM_L = V_{IL} - V_{OL}$; *High noise margin:* $NM_H = V_{OH} - V_{IH}$.
- Any valid output driven through noise up to NM still meets the next stage's input spec.
- Voltage-transfer characteristic (VTC): plot V_{out} vs. V_{in}, must avoid "forbidden" regions.

13 MOSFETs and CMOS Logic

- MOSFET = voltage-controlled switch with four terminals (gate, source, drain, bulk).
- *n-channel (NFET)* used in pull-down networks; conducts when $V_{GS} > V_{th}$.
- *p-channel (PFET)* used in pull-up networks; conducts when $V_{GS} < V_{th}$ (threshold negative).
- **CMOS inverter:** one NFET to ground, one PFET to V_{DD}, gates tied to input.
  - Input "0" → PFET on, NFET off → output = 1.
  - Input "1" → NFET on, PFET off → output = 0.
  - Both briefly on during transition → high gain, sharp switch.
- Complex gates: pull-up = complementary network of series/parallel PFETs to implement $\neg f$, pull-down = dual NFET network.

- **Timing:** t_{CD} (contamination delay) = lower bound from input invalid→output invalid; t_{PD} = upper bound from input valid→output valid.

14 Boolean Specification of Combinational Devices

14.1 Truth Tables

- A *truth table* exhaustively lists the output(s) of a device for every combination of its N binary inputs.
- There are 2^N rows. E.g. for $N = 3$ inputs $\{A, B, C\}$, we have 8 rows (000,001,...,111).
- Truth tables are unambiguous, but grow exponentially large—e.g. $N = 64$ would need 2^{64} rows!

14.2 Boolean Equations

- Rather than tabulate 2^N cases, we can write a formula using logical operations:

$$\text{AND}: X \wedge Y, \quad \text{OR}: X \vee Y, \quad \text{NOT}: \neg X, \quad \text{XOR}: X \oplus Y.$$

- Interpret $0 \leftrightarrow \mathsf{FALSE}$, $1 \leftrightarrow \mathsf{TRUE}$.
- Example: if a 3-input device has output $Y = 1$ on rows 2,4,7,8 of its truth table, then

$$Y = (\neg C \wedge \neg B \wedge A) \vee (\neg C \wedge B \wedge A) \vee (C \wedge \neg B \wedge A) \vee (C \wedge B \wedge A).$$

15 Sum-of-Products (SOP) Synthesis

- *Sum-of-products*: each minterm (product of literals) covers one row where output=1, then OR them together.
- Circuit recipe:
  1. Invert inputs as needed.
  2. Use one AND gate per product term.
  3. Use one OR gate to combine all product outputs.
- Library trade-offs: direct multi-input AND/OR vs. NAND+inverter chains for speed/area.

16 Karnaugh Maps and Logic Minimization

- **Karnaugh map (K-map)**: lay out 2^N cells in a Gray-code grid so adjacent cells differ in one bit.
- A *prime implicant* is a largest rectangular group (size 2^k) of 1's; each gives a simplified product term.
- To minimize:
  1. Fill K-map with 1's for output=1 rows.
  2. Circle all prime implicants (allow wrap-around).
  3. Choose a minimal cover (select enough prime implicants to cover every 1).
  4. Translate each implicant into a product of literals that remain constant over that block.
- This yields a minimal SOP with fewer gates and reduced glitch potential (lenient implementation if all primes are used).