FORMAL MODELLING OF CHANNEL-AND SOURCE-CODING ALGORITHMS FROM CODING THEORY

Thomas Michlmayr

March 23, 2026

State of the Art

- Today's digital world consists of transmitting and sending data e.g.:
  - phone calls
  - internet connection
  - messages
  - ...
- Why don't you experience problems with it very often? $\Rightarrow$ efficient coding and decoding of data.
- Proper usage of algorithms can prevent errors in communication.
- How do we achieve such comfort:
  1. Find proper algorithms for coding and decoding/transmitting messages. (Information Theory)
  2. Prove their correctness mathematically. (Mathematics)

Information Theory

- A communication system is defined as the path which data travels from a source to a destination.

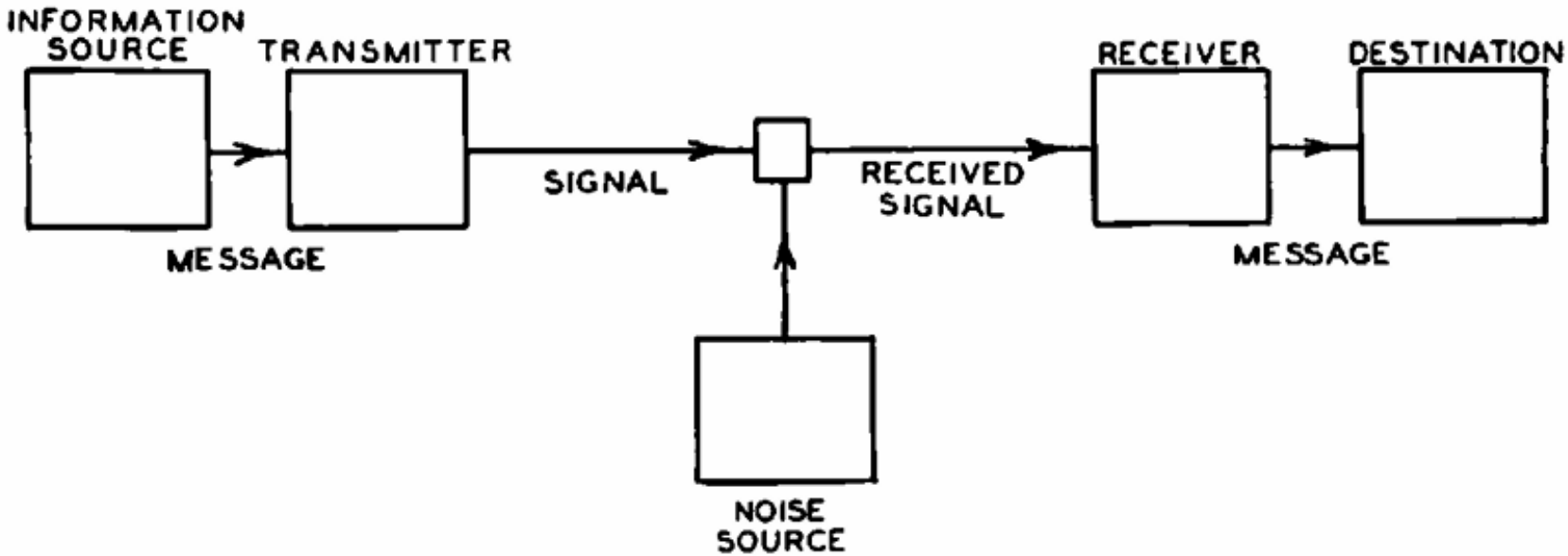

Fig. 1—Schematic diagram of a general communication system.

- In the real world, sometimes data gets corrupted (noise). This noise appears (typically) in the channel.

Information Theory

- To prevent total loss of data, different approaches can be made:
  - Increase channel stability against external influences (not part of my thesis)
  - Coding data such that errors can be detected and corrected (**E**rror **C**orrecting **C**odes) $\equiv$ Channel Coding
    - Deals with Channel capacity.
    - Is basically about adding redundancy.
  - Coding data such that it gets transmitted more efficiently (Compression) $\equiv$ Source Coding
    - Deals with Entropy.
    - Is basically about removing redundancy.

Example - Hamming Codes (Channel Coding)

We want to transmit the decimal number 12
$12_{dec} = 1100_{bin}$ so we have data bits
D1 = 1, D2 = 1, D3 = 0, D4 = 0 $\Rightarrow$ use (7,4) Hamming Code
which means 4 data bits, 3 parity bits, in total 7 bits
check positions Pi are always located at powers of 2, so Pi with
$i \in \{2^x \mid x \in \mathbb{N}\}$
form of Coded message: P1 P2 1 P4 1 0 0
Parity Bits Pi check different positions Parity (Pi = 0, if number of 1 (#1) bits is even; Pi = 1, if number of 1 (#1) bits is odd)
P1 checks positions 1, 3, 5, 7 $\Rightarrow$ P1 = 0
P2 checks positions 2, 3, 6, 7 $\Rightarrow$ P2 = 1
P4 checks positions 4, 5, 6, 7 $\Rightarrow$ P4 = 1
Coded Message: 0 1 1 1 1 0 0

Example - Hamming Codes (Channel Coding)

Coded Message: 0 1 1 1 1 0 0
let a noise occur $\Rightarrow$ Bit 5 flips to 0
$\Rightarrow$ Coded Message: 0 1 1 1 0 0 0
Receiver checks the message:
P1 checks positions 1, 3, 5, 7 $\Rightarrow \#1 = \text{odd} \Rightarrow S1 = 1$
P2 checks positions 2, 3, 6, 7 $\Rightarrow \#1 = \text{even} \Rightarrow S2 = 0$
P4 checks positions 4, 5, 6, 7 $\Rightarrow \#1 = \text{odd} \Rightarrow S4 = 1$
Construction of Syndrom: S4 S2 S1 $= 101_{bin} = 5_{dec} \Rightarrow$ indicates error at position 5 $\Rightarrow$ flip bit at position 5

extraction of data bits results in sent decimal number 12

Hamming Codes work fine for the occurence of **just one error**

Mathematics

- our system can be mathematically understood as functions such that:
  $\forall x, y : \mathit{somewhatcorrupted}(y, \mathit{encode}(x)) \Rightarrow \mathit{decode}(y) = x$
  with x as data
- $\mathit{somewhatcorrupted}(y, x)$ means, y is a version of x, that is not too corrupted.
- Thus, cheking the algorithms validity, first order logic (FOL) properties have to be proven.
- FOL on infinite domains is undecidable $\Rightarrow$ there is no algorithm which can determine the logical correctness of the algorithms for every input, hence this way of proving is not automatable.
  $\Rightarrow$ restriction to finite domains (models), **Formal Model Checking**

Mathematics - RISCAL

- Suitable tool for Formal Model Checking.
- Provides language where algorithms and logical properties can be described with logical formulas in FOL.
- Allows to automatical check the validity of models of fixed size via going through the whole state space.
- Even provides counterexamples when algorithms are not valid.

Algorithms I will use

- Channel Coding:
  - Parity Bit
  - Repetition Codes
  - Hamming Codes
  - Cyclic Redundancy Checks (CRC)
  - Reed Solomon Codes (optionally)
- Source Coding:
  - Run-Length Encoding
  - Golomb Coding
  - Huffman Coding
  - Shannon-Fanon Coding (optionally)
  - Lempel-Ziv-Welch (optionally)

Timeframe

- March: Prepare literature, start writing about "State of the Art"
- April: finish "State of the Art", start writing about Channel Coding, implement models in RISCAL
- May: finish Channel Coding, start writing about Source Coding, implement models in RISCAL
- June: finish Source Coding, write Conclusion, publish RISCAL files, Bachelor presentation