Codes

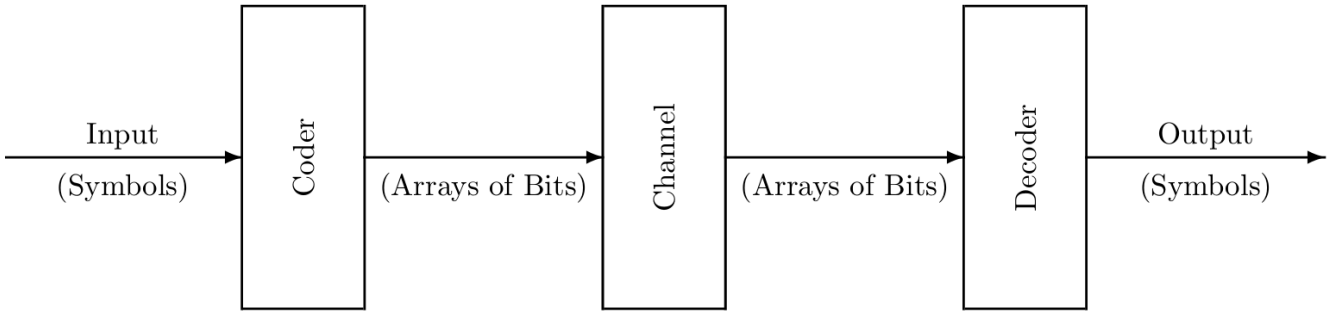

Figure 2.1: Simple model of a communication system

2.1 Symbol Space Size

If Symbol space size is **Finite**, we could use bits to represent them.

If Symbol space size if **Infinite** but **Countable**, we should find ways to deal with the **overflow**.

If Symbol space size if **Infinite** but **Uncountable**, we usually use **discretization(离散化)**. For example: if the numbers between 0 and 1 were the symbols and if 2 bits were available for the coded representation, one approach might be to approximate all numbers between 0 and 0.25 by the number 0.125, all numbers between 0.25 and 0.5 by 0.375, and so on.

2.2 Use of Spare Capacity

In many situations there are some unused code patterns, because the number of symbols is not an inte power of 2. There are many strategies to deal with this. Here are some:

- Ignore
- Map to other values
- Reserve for future expansion
- Use for control codes
- Use for common abbreviations

These approaches will be illustrated with examples of common codes.

2.2.1 Binary Coded Decimal (BCD)

1. BCD的基本概念

二进制编码十进制（Binary Coded Decimal, BCD）是一种用4位二进制数表示单个十进制数字（0-9）的编码方式。每个十进制数字对应一个唯一的4位二进制码，剩余6种组合（1010至1111）未使用。

- **目的**：简化十进制数的机器处理（如金融计算、数字显示），避免二进制与十进制转换的误差。

---

2. BCD编码表

根据表格内容，十进制数字与BCD码的对应关系如下：

十进制数字	BCD码（4位二进制）
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

未使用的组合：

- 1010、1011、1100、1101、1110、1111（共6种）。

---

3. 未使用组合的处理方法

在BCD系统中，若出现未使用的二进制组合（如1010），需设计策略应对可能的错误或异常。以下是两种常见方案：

(1) 忽略未使用的组合

- **策略**：
  - 解码器检测到无效码时，返回空值或触发错误信号。
- **优点**：严格区分合法与非法输入，便于错误检测。
- **缺点**：
  - 可能导致系统中断（如错误信号需额外处理）。
  - 无法从错误中恢复，需依赖外部纠错机制。

(2) 映射到合法值

- **策略**：
  - **直接映射**：将无效码统一转换为某个固定值（如全部转为9）。
    示例：1010（十进制的10）→ 9。
  - **部分修正**：根据特定规则调整部分位（如将最高位设为0，得到2-7）。
    示例：1010 → 0010（即2）。
- **优点**：系统可继续运行，避免中断。
- **缺点**：
  - 数据准确性降低（如错误被掩盖）。
  - 可能引入不可预测的行为（如错误传播）。

Genetic Code

将未使用的模式映射到合法值

2. **密码子长度与冗余设计**
   - 核苷酸有四种（A、C、G、T/U），因此：
     - 单个核苷酸最多指定4种氨基酸（不足）。
     - 两个核苷酸组合可指定16种氨基酸（仍不足，因蛋白质使用20种氨基酸）。
     - 最终采用**三位核苷酸（密码子）**，产生$4^3 = 64$种组合，远超所需的20种氨基酸。
   - **冗余性**：大多数氨基酸由多个密码子编码，增强了系统的容错性。例如：
     - 丙氨酸（Alanine）由所有以GC开头的四个密码子（GCU、GCC、GCA、GCG）编码，第三位核苷酸可突变而不影响功能（称为“摆动位”）。
     - 20种氨基酸中，有8种的第三位核苷酸是“无关位”，因其在转录过程中更易出错（称为**摆动效应**）。

1. 遗传密码的基本机制

- **编码单位**：
  蛋白质由20种氨基酸组成，每种氨基酸由三个核苷酸组成的**密码子（Codon）**编码。
  - **DNA/RNA结构**：
    - DNA含四种核苷酸（A、C、G、T），RNA含（A、C、G、U）。
  - **密码子长度**：
    三位核苷酸（如AUG）可生成43=6443=64种组合，远超所需的20种氨基酸，提供冗余容错能力。

---

2. 冗余与容错设计

- **简并性（Degeneracy）**：
  多个密码子编码同一氨基酸，降低突变影响。
  - **示例**：
    - 丙氨酸（Alanine）由以GC开头的四种密码子（GCU、GCC、GCA、GCG）编码，第三位核苷酸（“摆动位”）可突变而不改变氨基酸。
  - **生物学意义**：
    - 第三位核苷酸易受“摆动效应（Wobble Effect）”影响（RNA与tRNA配对时灵活性高），允许部分错配仍正确翻译。

---

3. 控制密码子的功能

- **起始密码子**：
  - **AUG**：编码甲硫氨酸（Methionine），同时标志蛋白质合成的**起始点。**
- **终止密码子**：
  - **UAA、UAG、UGA**：不编码任何氨基酸，标记蛋白质合成的**终止点**，确保不同长度蛋白质的正确截断。

---

4. 标准化编码的意义

- **通用性与效率**：
  - 所有蛋白质的合成由同一套分子机器（核糖体、tRNA等）完成，仅需根据DNA/RNA序列动态调整，无需为每种蛋白质定制合成机制。
- **信息存储代价**：

- 编码蛋白质的核苷酸序列原子数超过蛋白质本身，但标准化允许复用合成装置，整体效率更高。

---

5. 与人工编码的类比

- **控制字符**：
  - 类似计算机编码中的“起始位”和“终止位”，遗传密码通过专用密码子（AUG、UAA等）管理流程，而非仅传递数据。
- **冗余设计**：
  与纠错码（如海明码）类似，通过冗余提高容错性，确保生物过程稳健性。

Telephone Area Codes

The third way in which spare capacity can be used is by reserving it for future expansion.

2.2.4 IP Addresses

Another example of the need to reserve capacity for future use is afforded by IP (Internet Protocol)
For example: 127.0.0.1

2.2.5 ASCII

A fourth use for spare capacity in codes is to use some of it for denoting formatting or control operations
最常用的文本字符代码 ASCII（美国信息交换标准代码，在第 2.5 节中介绍）明确保留了 128 个代码中的 33 个用于控制，而只有 95 个用于字符。这 95 个包括英文字母表的 26 个大写字母和 26 个小写字母、10 位数字、空格和 32 个标点符号

2.3 Extension of Codes

经通用化的代码，往往会携带源自其原始应用场景的非预期偏差。有时，这些偏差仅让人觉得有趣，但在其他情况下，它们会使代码难以使用。
ASCII 携带的一个严重的偏差是使用回车（CR）和换行（LF）这两个字符来换行。在电传打字机中，移动纸张（连续卷筒纸）和将打印元件重新定位到左边缘分别由不同硬件实现。设计演变为 ASCII 的代码的工程师，肯定认为分别设置这些操作是有益的。但他们无法想象，随着 ASCII 应用于不同硬件环境，且无需像 CR 或 LF 那样分别控制打印位置时，给后来的使用者带来了多少麻烦。不同计算系统对此处理方式各异：Unix 使用 LF 换行并忽略 CR，Macintosh（至少在 OS X 之前）使用 CR 并忽略 LF，而 DOS/Windows 则两者都需要。这种不兼容性持续引发严重的困扰和错误。

2.4 Fixed-Length and Variable-Length Codes

在代码设计初期，需要决定是用相同位数的代码表示所有符号（定长编码），还是让部分符号使用比其他符号更短的代码（变长编码），这两种方案各有优势。

定长编码通常更易于处理，因为编码者和译码者事先都知道涉及的位数，只需设置或读取这些位的值。而定长编码还支持并行传输，即编码信息的各个位可以同时从编码者传送到译码者，例如通过多根导线传输电压。

变长编码则需要译码者确定一个符号的编码何时结束、下一个符号的编码何时开始。编码信息的串行传输通常与变长编码相关，在串行传输中，单根导线发送一连串的位，译码者必须判断一个符号的位何时结束、下一个符号的位何时开始。若译码者出现混淆或在编码流已经开始后才处理，就可能出现 “帧错误”。为消除帧错误，通常会在符号之间发送停止位，例如通过串行线路传输的 ASCII 码通常有 1 或 2 个停止位，一般取值为 0。这样，若译码者步调不一致，最终会在本应是停止位的位置发现 1，从而尝试重新同步。虽然理论上帧错误可能会持续很长时间，但在实践中，使用停止位的方法效果良好。

2.4.1 Morse Code