拟态语言技术手册

基础

之恪提案

2025 年 12 月 28 日

目录

1 基础概念

1.1 位

位是计算机中表示信息的基本单位，在二进制计算机中，它可以表示 2 种状态 (0 和 1)，以此类推，在 $\texttt{n}$ 进制中，它可以表示 $\texttt{n}$ 种状态 (0 到 $\texttt{n}-1$)。

1.1.1 进制

进制（进位计数制）是人为定义的通过基数规定进位规则的计数方法，其核心特征为"逢基数进一"。例如：

- 十进制：基数为 10，使用 0-9 共 10 个数码，逢十进一
- 二进制：基数为 2，使用 0 和 1 共 2 个数码，逢二进一
- 八进制：基数为 8，使用 0-7 共 8 个数码，逢八进一
- 十六进制：基数为 16，使用 0-9 和 A-F 共 16 个数码，逢十六进一

可以发现一个规律：$\texttt{n}$ 进制逢 $\texttt{n}$ 进 1，因此它的数码中不含 $\texttt{n}$。例如十进制没有值为 10 的个位数，二进制没有值为 2 的个位数。

1.1.2 基于进制的计数系统

在日常生活中，我们使用十进制计数。在十进制的计数系统中，我们把十进制数字的各个位置分为个位、十位、百位、千位...。我们来解析一下，已知基数为 10，假设一个三位数的个位数字为 $\texttt{a}$，十位数字为 $\texttt{b}$，百位数字为 $\texttt{c}$。那么它的值可以分解成 $\texttt{a}+10\texttt{b}+100\texttt{c}$，也就是 1 倍的 $\texttt{a}$、10 倍的 $\texttt{b}$、10 倍的 10 倍的 $\texttt{c}$ 求和。换言之，即 $10^0\texttt{a}+10^1\texttt{b}+10^2\texttt{c}$，其中可以发现一个规律：如果从 0 开始计数，那么第 0 位就是个位，权重为 $10^0=1$；第 1 位就是十位，权重为 $10^1=10$；第 2 位就是百位，权重为 $10^2=100$。

现在把视角换为二进制，二进制计数系统的基数为 2，因此第 0 位的权重为 $2^0=1$；第 1 位的权重为 $2^1=2$；第 2 位的权重为 $2^2=4$...。

现在你应该可以理解，对于任意的 $\texttt{n}$ 进制，它的计数系统是如何运作的了。

1.1.3 存储空间中的计数系统

在二进制计算机的存储空间中，一个位可以表示 0 和 1，可以发现它和二进制计数系统的位的表示范围相同，因此，我们可以用计算机中的位代表计数系统中的位。例如，假设我们可以控制计算机中的 $\texttt{n}$ 个位，那么我们就可以通过编码这 $\texttt{n}$ 个位来表示一个 $\texttt{n}$ 位的二进制数。通常，这 $\texttt{n}$ 个位在计算机中是顺序存储的，你可以把位想象为一个小方格，它们在存储空间中是紧挨着的，排成一个长条。

现在你应该可以理解，计算机是如何表示数字的了。

1.2 字节

字节是计算机中由多个位组成的一个单位，例如，在目前的计算机中，我们规定 8 个位组成一个字节。假设进制为 $\mathtt{n}$，一个字节由 $\mathtt{m}$ 个位组成，那么一个字节的状态数为 $\mathtt{n}^{\mathtt{m}}$，表示范围就是 $[0, \mathtt{n}^{\mathtt{m}} - 1]$。

1.3 地址

地址是一个数字，每个地址都代表了计算机中某一个字节的位置。例如，地址 $\mathtt{n}$ 和 $\mathtt{n}+1$ 指向两个在存储空间中相邻的字节。

因此，如果我们要准确地在计算机中表示一个唯一的位的位置，可以使用这个位所在的字节的地址，以及它在这个字节中是第几个位来表示。

1.4 字长

字长是衡量计算机性能的一个关键指标，它决定了计算机一次操作所能处理的数据量。例如，64 位且字节为 8 位的计算机的字长就是 8 个字节，它决定了这台计算机一次操作可以处理 8 字节的数据。

因此，此计算机中大部分的操作至少都可以一次性处理 8 个字节，例如加减乘除等。

2 数胞

开头：我们要解决什么问题？

在前面的章节中，我们学习了比特、字节等存储单元。我们发现，无论是哪种进制的计算机，其存储单元都可以被概括为两个核心属性：

1. 该单元所能呈现的**所有可能状态的数量**（$\mathtt{n}$）。
2. 该单元在某一时刻所处的**具体状态值**（$\mathtt{m}$）。

为了用一种统一、简洁的数学语言来描述计算机中所有的存储空间，我们引入了“数胞”（Numerical Cell）这一抽象概念。它将帮助我们跳出二进制、十进制等具体进制的限制，从更高层面理解存储的本质。

2.1 数胞的定义与表示

一个数胞（NC）可以由一个二元组完整定义：

$$NC \equiv (\mathtt{n},\ \mathtt{m})$$

其中：

- `n`：表示该数胞的**状态数**（Number of States）。它一般是一个大于 1 的自然数，决定了该存储单元能表示多少种不同的情况。它本质上对应了前文所述的“进制”概念。
- `m`：表示该数胞的**值**（Value）。它是该存储单元在当前时刻的具体状态，是一个自然数，且必须满足 $0 \le \mathtt{m} < \mathtt{n}$。

2.2 数胞的简写符号

为了书写方便，我们定义了一套简写符号：

- NC：数胞（Numerical Cell）的通用缩写。
- $NC_{\mathtt{n}}$：表示所有状态数为 `n` 的同一类型数胞的集合。它强调了存储单元的“类型”或“容量”。
- $NC_{\mathtt{n}}(\mathtt{m})$：表示一个状态数为 `n`，且当前值为 `m` 的特定数胞。这是最完整的表示形式，同时包含了存储单元的属性和当前状态。

示例：

- $NC_2(1)$：表示一个状态数为 2（即二进制）、当前值为 1 的数胞。这其实就是我们熟悉的一个比特（Bit），其值为 1。
- $NC_{256}(65)$：表示一个状态数为 256、当前值为 65 的数胞。这恰好可以表示一个值为 65 的字节（Byte），因为一个 8 位二进制字节有 $2^8 = 256$ 种状态。
- $NC_{10}(7)$：表示一个状态数为 10（即十进制）、当前值为 7 的数胞。这可以想象为一个十进制的基本存储单元。

2.3 数胞的抽象意义

“计算机中所有的存储空间都可以抽象为一个数胞。”这句话的含义是：无论存储空间的实际物理实现如何，我们都可以用数胞的二元组模型 $(\mathtt{n}, \mathtt{m})$ 来刻画它。

- **若一个存储单位可以表示 `N` 个状态**：这意味着它的“类型”是 $NC_{\mathtt{N}}$。
- **且其当前值为 `M`**：这意味着它的当前状态是 `M`，且 $0 \le \mathtt{M} < \mathtt{N}$。
- **那么它可以被数胞 $NC_{\mathtt{N}}(\mathtt{M})$ 表示**：这个数胞的表示完全捕获了该存储单元的核心信息。

数胞的抽象威力在于其通用性。它不关心底层是二进制电路、三进制器件还是其他任何物理实现，它只关心逻辑上的状态数量和一个具体的状态值。这使得我们可以在统一的框架下讨论不同架构的计算机存储问题。

2.4 状态数公式的再现

一个由 k 个 $NC_{\mathtt{n}}$ 类型的数胞连续组成的存储空间，其总状态数正是我们熟悉的公式：

$$\text{总状态数} = \mathtt{n}^{\mathtt{k}}$$

这个公式解释了为什么一个由 8 个 NC_2（比特）组成的字节 ($\mathtt{n} = 2, \mathtt{k} = 8$) 有 256 种状态 ($2^8$)，也解释了一个由 2 个 NC_{10}（十进制单元）组成的存储空间有 100 种状态 (10^2)。