计算理论导引笔记

2025

by kiwiizzz,xhkzdepartedream

Contents

§1. 计算理论

§1.1. 引入

§1.1.1. 图灵机

Definition 1.1.1.1 (图灵机).
一台 k-带图灵机 $\mathbb{M}$ 是一个**三元组** (Γ, Q, δ) ,其中:

1. 有限符号集 Γ , s.t. $\Gamma \supseteq \{0, 1, \square, \triangleright\}$;
2. 有限状态集 Q, s.t. $Q \supseteq \{q_{start}, q_{halt}\}$;
3. 迁移函数 $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{L, S, R\}^k$.

$\mathbb{M}$的第一条带子是只读的输入带,剩下的都是工作带;最后一条带子是输出带.

> This course is about *classifying and comparing* problems by the amount of resource necessary to solve them.
>
> — Turing, A. M.

解读:

1. 有限状态集Q 是机器所有可能“心理状态”的集合，比如“正在读输入”“正在计算”“准备输出”等。

必须包含: q_{start}开始状态，机器一启动就处于这个状态; q_{halt}停机状态，一旦进入这个状态，机器就停止运行，不再执行任何操作。

1. 你可以在Γ, Q中自己定义别的奇妙的符号,但必须包含以上特别指出的符号.

2. 迁移函数输入:当前状态 + k 个读写头各自看到的符号(每条带子一个符号，共 k 个)
   输出：下一个状态(Q 中的一个)；要写到每条带子当前格子的新符号（k 个符号 $\to \Gamma_k$)；每个读写头下一步怎么移动：L, S, R 分别表示读写头向左、右或者保持不动.

Example.
图灵机的初始化:

1. $\triangleright$ 在每条带子的最左端；
2. 所有工作带的格子里都放空符号 $\square$;
3. 输入带除了输入之外的地方也是空符号.

我们自然的会把 $(q, a_1, ..., a_k) \xrightarrow{\delta} (q', a_2', ..., a_k', A_1, ..., A_k)$ 称为一条**指令**. 其中 $A_i \in \{L, S, R\}$ 即一条移动;根据定义,a_1 不需要修改,因为是只读的.

§1.1.2. 格局

下面来看**格局**.

Definition 1.1.2.1 (格局).
$\mathbb{M}(x)$ 表示预制输入 x。计算的任意时刻 t 的**格局** $\sigma_t = (q, \kappa_2, ..., \kappa_k, h_1, ..., h_k)$ 是一个 $2k$ 元组,其中 κ_i 是工作带内容,$h_i \in \mathbb{N}$ 是读写头位置.

初始格局:$(q_{\text{start}}, \varepsilon, \varepsilon, \cdots, \varepsilon, 0, 0, \cdots, 0)$

种植格局:$(q_{\text{halt}}, \kappa_2, ..., \kappa_k, h_1, ..., h_k)$,只要求$q_{\text{halt}}$.

解读:
1. t时刻的格局能反映这一时刻的图灵机状态,包含当前状态q,$2 \sim k$条纸带的状态$\kappa_2 \sim \kappa_k$,每条纸带上读写头的位置h_i.
2. 初始格局是唯一的. 如果达到了终止格局,格局之间的传递被称为**计算路径**,详见 P6.
3. 时间函数记作 $T(n)$, $n = |x|.T(n) \geqslant n$,这是因为我们假定图灵机必须先完整的读一遍输入.
4. 可以用“快照”来理解“格局”，但“快照”在后面的部分仍有定义，不要混淆。

§1.1.3. 问题与语言

Definition 1.1.3.1 (问题与语言).
1. 一个函数 $f : \{0,1\}^* \to \{0,1\}^*$ 被称为一个**问题**.
   - 如果对每个 $x \in \{0,1\}^*$，图灵机 $\mathbb{M}$ 满足 $M(x) = f(x)$，则称 M **计算**或**求解**函数 f.
   - “$M(x) = y$”表示“图灵机 $\mathbb{M}$ 在输入带预装 x 的情况下停机，且输出带上写有 y”.
2. 一个函数 $d : \{0,1\}^* \to \{0,1\}$ 被称为一个**判定问题**.
   - 如果图灵机 M 计算 d，则称 M **判定**d.

一个集合 $L \subseteq \{0,1\}^*$ 被称为一个**语言**. 其中 L^* 表示 L 中元素的有限串集合.

- 如果图灵机 M 判定如下特征函数:

$$L(x) = \begin{cases} 1, \text{如果 } x \in L \\ 0, \text{如果 } x \notin L \end{cases}$$

那么称 $\mathbb{M}$ 接受语言 L，或者 L 是可判定的.

- 一个判定问题 $A \subseteq \{0,1\}^*$ 的补问题 $\overline{A}$ 定义为 $\{0,1\}^* \setminus A$.

解读: 问题就是函数,函数就是问题. 参考书例子: 回文串 (朴素, P8).

一个问题是**图灵机可解的** $\Leftrightarrow$ 有一台图灵机计算它.

Theorem 1.1.3.1 (思考题).
为以下函数设计图灵机:

1. $s(x) = x + 1$. 用进位处理实现计数器.
2. $u(x) = 1^x$. 附加 1^x 限制运行步数.
3. $e(x) = 2^x$. 输出 2^x.
4. $l(x) = \log x$. 若 $x = 2^k$,输出$|x| + 1$,否则输出$|x|$.

Proof.

1. $s(a) = +1$。先找到串的末尾，然后做如下操作:
   1. 如果当前位是 1，将其翻转为 0，然后向左移动一位;
   2. 如果当前位是 0，将其翻转为 1，然后停机;
   3. 如果是 $\triangleright$，改写为 1，然后停机。
2. $u(a) = 1^n = \underbrace{11\cdots11}_{n\text{个}1}$。对输入端做减法，然后减 1 个就输出一个 1。
3. $e(a) = 2^n$。用第二条给出 1^n，然后对这个一元串，读的第一个写 1，读的后续所有都写 0。
4. $l(a) = \log(a)$。先判断 x 的二进制有几位，有几位就输出几个 1，成为串 L; 然后判断 x 是不是 2 的幂次，如果第一个是 1 就往右挪，如果碰到了另一个 1 就是 N，否则是 Y; 最后，如果是 Y 就在 L 后补一个 1，如果是 N 就不做，最后，把 L 变成二元串。

□

§1.2. 时间可构造性

Definition 1.2.1 (时间可构造性).
一个函数 $T : \mathbb{N} \to \mathbb{N}$ (且 $T(n) \geq n$) 被称为是 时间可构造的 (Time Constructible)。

- 如果存在一台图灵机 M，它能在 $O(T(n))$ 时间内计算并输出 $T(n)$ 的二进制表示 (即计算函数 $1^n \mapsto \lfloor T(n) \rfloor$)。

一个函数 $T : \mathbb{N} \to \mathbb{N}$ (且 $T(n) \geq n$) 被称为是 完全时间可构造的 (Fully Time Constructible)。

- 如果存在一台图灵机 M，它在接收输入 1^n 后，会在**恰好** $T(n)$ 步后停机。

解读:时间可构造性是指,$T(n)$本身可以被一个图灵机在接收1^n(表示 n 这个数)的输入后,在有限步($O(T(n))$)内,输出n的二进制表示.

若在图灵机里面加入一个不停向右移动一格、到达规定位置后停机的条带,就得到了一个**时钟图灵机** $\mathbb{T}$作为计时器,用以限制另一台图灵机($\mathbb{M}$)的运行时间,强制它在指定步数内停止.

Definition 1.2.2 (硬连接)**.**
我们可以构造一台新的图灵机 $\mathbb{M}'$，它(利用时间函数是$T(n)$的时钟图灵机$\mathbb{T}$)通过以下方式为 $\mathbb{M}$ 的计算强制设定一个时间上限:

1. 在输入 x 上, $\mathbb{M}'$ 首先计算出步数上限 $k = T(|x|)$。(由于 T 是时间可构造的, 此步骤可在 $O(T(|x|))$ 时间内完成)。
2. $\mathbb{M}'$ 模拟 $\mathbb{M}$ 在输入 x 上的运行，最多 k 步。
3. 如果 M 在 k 步内停机，M' 就停机并返回 M 的结果。如果 M 在 k 步内没有停机，M' 会强制停机并进入**拒绝状态**。

具体的构造如下:

$$\mathbb{M} = (Q_0, \Gamma_0, \delta_0), \mathbb{T} = (Q_1, \Gamma_1, \delta_1), \mathbb{M}' = (Q, \Gamma, \delta)$$
$$Q = Q_0 \times Q_1, \text{assuming } (q, q^1_{\text{halt}}) = (q^0_{\text{halt}}, q) = q_{\text{halt}}$$
$$\Gamma = \Gamma_0 \times \Gamma_1$$
$$\delta = \delta_0 \times \delta_1$$

解读: 构造一台集成的图灵机$\mathbb{M}'$,前k_0条给$\mathbb{M}$用,后k_1条给$\mathbb{T}$用,用笛卡尔积来定义$\mathbb{M}'$的状态.新机器的停机条件为:$\mathbb{M}$和$\mathbb{T}$其中有任何一台停机.

一个图灵机被称为**神游的**，如果读写头的逻辑是确定的。

§1.3. 通用图灵机

Definition 1.3.1 (通用图灵机)**.**

1. 一个转移规则 $(p, a, b, c) \to (q, d, e, R, S, L)$ 可以被编码为，比如说:

   $$001 \dagger 1010 \dagger 1100 \dagger 0000 \dagger\dagger 011 \dagger 1111 \dagger 0000 \dagger 01 \dagger 00 \dagger 10$$

   ,$\dagger$是不同字符之间的分隔符,$\dagger\dagger$表示状态编码和转移方向之间的间隔.
2. 一个转移表可以被编码为一个具有如下形式的字符串:

   $$\ddagger \langle \text{rule}_1 \rangle \ddagger \langle \text{rule}_2 \rangle \cdots \ddagger \langle \text{rule}_m \rangle \ddagger$$

其中，二进制表示可以通过以下映射获得:

$$0 \to 01,$$
$$1 \to 10,$$
$$\dagger \to 00,$$
$$\ddagger \to 11.$$

解读:
就像程序可以作为数据被另一个程序读取和执行，这一部分的目标是：把一台图灵机的所有信息 (状态、符号、转移规则) 压缩成一个二进制字符串，这样它就可以被其他图灵机读取和模拟。
编码的一些性质:

- 每一台图灵机都可以表示为 01 编码的字符串,且编码是无限的,因为$0^i\sigma 0^j \iff \sigma$;

- $\forall\Sigma\in\{0,1\}^*$,它都代表着一个图灵机.(即使有一些无意义的串,也将其强行联系某个垃圾图灵机.比如0^n根本编码不出$\ddagger$,所以他是**非法的**;基于此,我们把所有的非法串映射到一个特定的$\mathbb{M}_0$)

我们可以接着考察图灵机的枚举.根据上面的假定,我们可以建立一个双射$f:\{0,1\}^*\longrightarrow\mathcal{TM}$,$\mathcal{TM}$是所有图灵机构成的集合.我们把它称作对图灵机的枚举.

用

$$\varphi(x)\triangleq\begin{cases}y, & \mathbb{M}_{i(x)}=y\\ \uparrow, & \mathbb{M}_{i(x)}\uparrow\end{cases}$$

来编码可计算的图灵机.

Theorem 1.3.1 (枚举定理,以及更多)**.**
存在一个通用图灵机 $\mathcal{U}$，它使以下陈述为真:

1. 对于所有的 $x,\alpha\in\{0,1\}^*$，有 $\mathcal{U}(x,\alpha)\simeq M_\alpha(x)$。等价关系的定义是,要么他们都停机且输出相同,要么他们都在运行.
2. 如果 $M_\alpha(x)$ 在 $T(|x|)$ 步内停机，那么 $\mathcal{U}(x,\alpha)$ 在 $cT(|x|)\log T(|x|)$ 步内停机，其中 c 是一个关于 α 的多项式。

- 具有 $\mathcal{O}(T(n))$ 时间放大的版本出现于 1965 年。
- 当前版本发表于 1966 年。

解读：存在一台通用图灵机，它可以模拟任何一台图灵机。这是说：对于任意的输入，上述两台图灵机的行为完全等价（产生的输出完全相同或者同时永不停机）；代价是一点额外的时间。

Proof. 我们来构造一个满足的图灵机。完整的证明在 P13-15，给出思路如下:

1. 我们先把 U 的主工作带分成 k 层，用一条工作带虚拟的模拟 k 条带(用存 k 元组的方法来解决);我们把 0 处的位置记作读写头读取的东西,然后我们把读写头的移动模拟为带子的反向移动.
2. 存放冗余信息（用 $\times$ 表示），把主工作带划分为以 2^i 为长度的多个区间，如图所示;

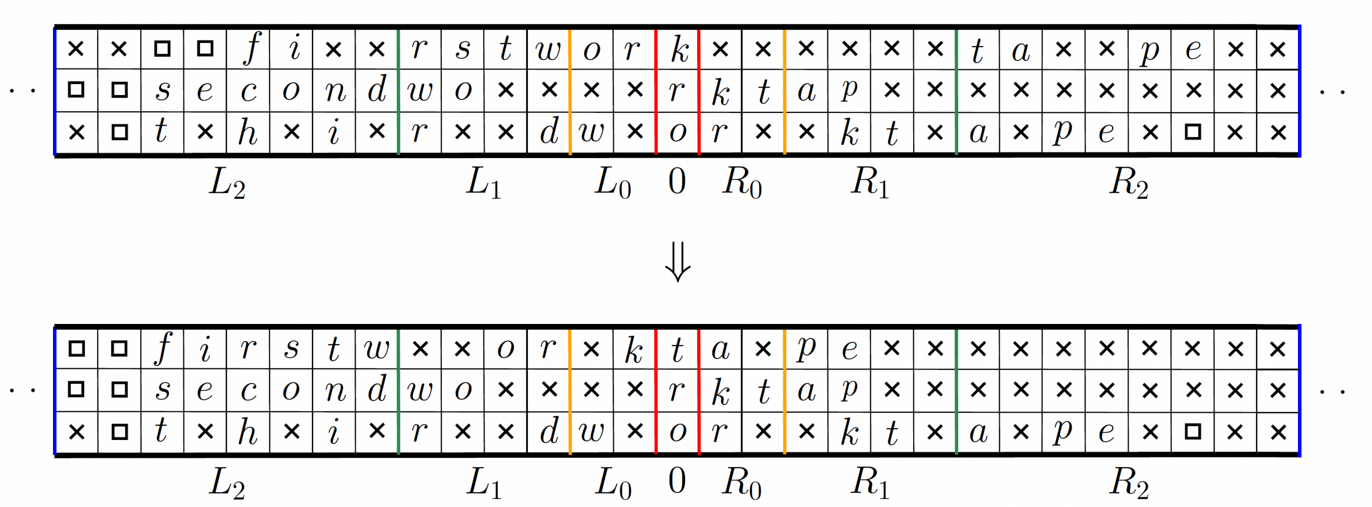

Figure 1: 证明图

3. 填充的规则如下: L_i,R_i 上填充的有效信息加起来是 2^{i+1}，且单个是只有满、半满、空 3 种状态。填充这个需要时间复杂度 $O(T\log T)$。

4. 来考察移动。只考虑左移。找到 i，s.t. $R_0, ..., R_{i-1}$ 全空，然后把 R_i 中的 2_i 个元素把前面 $i-1$ 个全部填充成半满，同时替换 0 层；然后，原有的 L_{i-1} 中的有效信息填入 L_i 中，以此类推。图见上。我们可以通过$\mathcal{U}$的第二条工作带来复制碰到的第一个非冗余信息,然后在区间大小的线性时间内完成复制工作.这里的时间复杂度是

$$\sum_{i=0}^{\log T(n)} \left(2^i \cdot \frac{T(|x|)}{2^i}\right) = O(T(n)) \cdot \log(T(n)).$$

只需证明 $k \le c(x)$。由于 $k \le |x|$，这是显然的. □

解读:

- 我们还是要用 01 串来编码符号集合.
- In other words, once a shift with index i is performed, the next $2^i - 1$ shifts of that parallel tape only involves indexes less than i. yyl 补全这些
- 这句话说明了,调整一次后,调整的内容里面垃圾是足够多的,我们的下面的操作不需要太复杂,因为有足够多的垃圾等我们搬空.这种设计方式大大减少了复杂性.

我们称一个图灵机是**健忘的**，如果在计算的时刻我们不考察输入了什么，而是考察输入的长度和操作了多少步。

修改 U 的定义：预先将所有 L_i, R_i 区间半满，然后引入计数器模拟步数。如果第 i 位从 0 变成 1，扫描 0 -i 的区间，把 $L_j, R_j, j \in \{0, 1, 2, ..., i-1\}$ 置为半满。这样，他就是健忘的，那我们有如下推论:

Proposition 1.3.1 (健忘图灵机)**.**
一些推论 thm:consequences 设 L 可在 $T(n)$ 时间内被图灵机判定，那么，$\exists C T(n) \log T(n)$ 时间内可判定 L 的健忘图灵机，其中 C 是常数。

设 $f \in \mathsf{TIME}(T(n))$，有双读写带的图灵机在 $O(T(n) \log T(n))$ 时间内计算 f，有单读写带的图灵机在 $O(T(n^2))$ 时间内计算 f。

§1.4. 对角线方法

不存在一段程序(图灵机)，使得其能够计算所有函数。可以使用对角线法以及通用图灵机构造出这样的反例。我们的目标是找到一个无法被图灵机计算的函数。

Problem 1.4.1 (decidable)**.**
UC 问题:

$$\mathcal{UC}(\alpha) = \begin{cases} 0, \text{ if } \ \mathbb{M}_\alpha(\alpha) = 1 \\ 1, \text{ otherwise} \end{cases}$$

停机问题:

$$\mathcal{Halt}(\alpha, x) = \begin{cases} 1, \text{ if } \ \mathbb{M}_\alpha(x) \downarrow \\ 0, \text{ otherwise} \end{cases}$$

它们都不能被判定。

Proof.

1.$\mathcal{UC}$需要循环嵌套.假定$\mathbb{UC}$可以判定$\mathcal{UC}$,那么,

$$\mathbb{UC}(\llcorner \mathbb{UC} \lrcorner) = 0 \iff \mathbb{UC}(\llcorner \mathbb{UC} \lrcorner) = 1$$

(直接代入 UC 的定义就可以得到这里的等价关系,从而导出了矛盾). (张小皓提出了这个反例在互换$\mathbb{UC}$里 0 和 1 就失效了,但事实上两者都不可判定.)

2.利用反证，假设存在这样的 $\mathbb{M}_H$，那么我们给出可以判断 $\mathcal{UC}$命题的图灵机的构造:

$$\mathbb{M}_H(\alpha) = \begin{cases} 0, \ \mathbb{M}_H(\alpha,\alpha) = 1 \wedge \mathbb{M}_\alpha(\alpha) = 1 \\ 1, \ \text{otherwise} \end{cases}$$

这便给出了一个矛盾。 □

解读: 1.否定性命题的一般证明方法就是利用反证法,其中对角线方法给了很好的例子.

§1.5. 加速定理

加速定理的动机在于思考:对于一个问题,有没有最好的算法解决它? 我们现在知道,结论否定(**Blum's Speedup Theorem**).为了证明他,我们不证明他.转而看向线性加速定理.

Theorem 1.5.1 (Linear Speedup(线性加速定理)).
设图灵机$\mathbb{M}$在$T(n)$步内判定L。对任意$\varepsilon > 0$,有 在图灵机$\mathbb{M}'$，$\mathbb{M}'$能在$\varepsilon T(n) + n + 2$步内判定L.

Proof. 我们来构造这个$\mathbb{M}'$.

- $\mathbb{M}'$的符号集$\Gamma' \supseteq \Gamma^m$,并且在$n+2$步内把原来的符号压缩$m$倍.其中 2 指的是从$\triangleright, \square$浪费的两步.
- $\mathbb{M}'$的读写头回正,需要$\frac{n}{m}$步.
- $\mathbb{M}'$的读写头模拟 m 步操作,至多需要 5 步.
- 则

$$T'(n) \leqslant n + 2 + \frac{n}{m} + \frac{5}{m}T(n) \leqslant n + 2 + \frac{6}{m}T(n)$$

- 我们取$m = \frac{6}{\varepsilon}$即可.

□

解读:

1.你总能把原来的运行时间压缩到任意小的比例(ε很小)，只需要再加一点“启动开销”$(n+2)$。核心思想是: 让新机器 $\mathbb{M}'$ 一次“看多格”，而不是像老机器 $\mathbb{M}$ 那样一格一格慢慢挪。

2.$\mathbb{M}'$需要:

- 读取当前“超级格子”的内容（包含 m 个原始字符）$\to$ 1 步
- 根据 $\mathbb{M}$ 的状态和当前字符，查表决定下一步动作 $\to$ 1 步
- 修改“超级格子”中的某个字符（如果需要）$\to$ 1 步
- 决定读写头往左/右移动:
- 如果向右移出当前“超级格子”，就要跳到下一个“超级格子”，并调整内部位置 $\to$ 1 步；如果向左移出当前“超级格子”，同理 $\to$ 1 步
- 更新状态 $\to$ 1 步

所以，最多需要 5 步 来完成一次“模拟 m 步中的一步”的操作。

当$T(n) \gg n$我们有一个类似小 o 的思想:

Proposition 1.5.1.
设图灵机$\mathbb{M}$在$T(n) \gg n$步内判定L.(或者我们记作,$T(n) = \omega(n)$)对任意$\varepsilon > 0$,有 在图灵机$\mathbb{M}'$，$\mathbb{M}'$能在$\varepsilon T(n)$步内判定L.(这就是说，线性项被忽略了。)

证明见作业§2.4.

§1.6. 时间复杂性类

Definition 1.6.1 (时间复杂性类).
A decision problem $L \subseteq \{0,1\}^*$ is in $\mathbf{TIME}(T(n))$, if there exists a $\mathbb{TM}$ that accepts L and runs in time $cT(n)$ for some $c > 0$.

解读:
$\mathbf{TIME}(T(n))$是一个集合：所有能在$T(n)$时间内被图灵机解决的问题的集合.

基于线性加速定理,我们不能定义问题的复杂性类,只能定义解决问题的时间的复杂性类.

我们先提出了 $\mathbf{P}$。一个**复杂性类**是一个模型无关的问题类. 我们把所有具有多项式时间算法的问题认为是同一类，即令

$$\mathbf{P} = \bigcup_{c \geq 1} \mathbf{TIME}\left(n^c\right)$$

以及

$$\mathbf{EXP} = \bigcup_{c \geq 1} \mathbf{TIME}\left(2^{n^c}\right)$$

然后我们提出了一个“明显的定理”（证明他！)。

Theorem 1.6.1 (时间复杂性类).
我们称 $\mathbf{coT} = \left\{\bar{A} \mid A \in \mathbf{T}\right\}$,$T$是一个时间复杂性类.$\mathbf{coT} \subseteq \mathbf{T} \Leftrightarrow \mathbf{coT} \supseteq \mathbf{T} \Leftrightarrow \mathbf{coT} = \mathbf{T}$

Proof. 有人托梦给我证明了.——YYL

□

§1.7. 非确定图灵机

§1.7.1. 非确定图灵机的定义及其时间定义

非确定图灵机的引入是为了解决**验证问题(Validation Problem)**.它不是物理上可实现的,但是他很好玩.

What nondeterminism provides is the power of **guessing**.

非确定图灵机 $\mathbb{N}$ 指的是有**两个**迁移函数的图灵机，它可以不确定地使用两者的策略之一。换言之，每一步可以有多个选择。从初始状态出发，其所有的可能路径会像二叉树不断向下延伸(类似《银河系漫游指南》的无尽概率跃迁方程)，每条路径可能终止，也可能不停机。非确定并不是随机。

Definition 1.7.1.1 (非确定图灵机的接受与拒绝，及其时间定义).

1. 设非确定图灵机 $\mathbb{N}$，它的每条**计算路径**(也就是一个$\delta_i, i \in \{0,1\}$的序列)试图构造一个存在性证明。如果构造成功，输入 1 并称这个终止格局叫**接受格局**；类似的定义**拒绝格局**。当输入 x 的时候，若 N 有一条计算路径终止于接受格局，称 $\mathbb{N}$ **接受** x，并记作 $\mathbb{N}(x) = 1$；类似定义**拒绝**。
2. 对于每一个输入 x，以及每一个“非确定性选择序列”，机器 $\mathbb{N}$ 必须在 $T(|x|)$ 步内停机（到达 q_h)。

§1.7.2. $\mathbf{P}, \mathbf{NP}, \mathbf{EXP}, \mathbf{NEXP}$问题

类似 $\mathbf{P}$，我们有 $\mathbf{NP}, \mathbf{NEXP}$。有一个明显的结论（P27）

Theorem 1.7.2.1.

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP} \subseteq \mathbf{NEXP}.$$

Proof. 第一个和第三个包含关系很显然；中间的包含关系是说，非确定性图灵机（NTM）在多项式时间内能解决的问题，可以用确定性图灵机在指数时间内模拟出来。根据非确定性图灵机的定义，这也是可以理解的。 □

§1.7.3. 快照

类似的考察通用性，我们给出**快照**的定义：

Definition 1.7.3.1 (快照).
一个 $k+1$ 元组 $(q, a_1, ..., a_k) \subseteq Q \times \Gamma \times \cdots \times \Gamma$。它包含了读写头指向的符号，但是不包含读写头位置和剩余工作带的内容.

解读:
1. 基于一个给定的图灵机,快照的大小是恒定的,而不像格局的长度是$O(T(|x|))$.

§1.7.4. 通用非确定图灵机

类似前文,我们可以定义一个“通用”的非确定图灵机$\mathbb{V}$.
1. 它猜测$\mathbb{N}_\alpha(x)$的一个终止于接受格局的快照序列和一个同等长度的 0-1 串,后者表示$\mathbb{N}_\alpha$在计算时做的非确定选择，即δ_0, δ_1的一个.猜测快照序列时，$\mathbb{V}$只考虑输入带上的符号，忽略工作带上的符号；工作带上的符号通过猜测得到.
2. $\mathbb{V}$需要多一条带来记录$\mathbb{N}_\alpha$的实际读写过程.对每一条工作带,$\mathbb{V}$对之前猜测到的 01 串中的合法的快照带回去验证是否成功.注意,这个验证是逐带进行的.

解读:

1.$\mathbb{V}$的代价是它都不一定停机,因为我们没办法确定快照序列的长度,也不能确定它完全停机;有一个办法是利用时钟图灵机$\mathbb{T}$来强制让他停止.

2.

3. 验证的进行:首先,$\mathbb{N}$也需要写入操作数这一个过程,我们的快照也记录下了这个信息;读写头的移动被δ_i确定了;读的什么内容被快照决定了(这就是确定了读/写了什么内容).

§1.8. 命题 & 谓词逻辑

关于命题和谓词逻辑的基本知识参考离散数学.来看在图灵机下的新的定义.

Definition 1.8.1.
可满足性问题是所有可满足的合取范式的集合,记作 **SAT**.k-合取范式中,每个语句最多含有k个字;这些合取范式的集合被记作 kSAT.

Proposition 1.8.1.
$\mathbf{SAT} \in \mathbf{NP}$.

Proof. 给定合取范式$\phi(x_1, x_2, \cdots, x_n)$,一个多项式时间的非确定图灵机可以猜测一个指派,然后验证之. □

为了简化问题,我们规定谓词逻辑的个体域是布尔域$\{0,1\}$,并简写$\exists x_1 \exists x_2 \cdots \exists x_k$为$\exists x_1 x_2 \cdots x_k$(当然,$\forall$也这么做).

Definition 1.8.2 (QBF).
如下形式的前束范式被称为**量化布尔公式**:

$$Q_1 x^1 Q_2 x^2 \cdots Q_k x^n \cdot \phi(x^1, \cdots x^n)$$

其中,x^i为**一串变量**,Q_i为$\forall, \exists, \forall, \cdots$或者$\exists, \forall, \exists, \cdots$.

问题 QBF 是所有永真的量化布尔公式的集合.当然也有 kQBF.

§1.9. 很多定理

我们来讨论对不同的函数 f, g, $\textbf{TIME}(f(n)) \subseteq \textbf{TIME}(g(n))$ 是严格的。我们给出三个定理。

Theorem 1.9.1 (时间谱系定理).
若函数 f, g 是时间可构造的,而且 $f(n)\log f(n) = o(g(n))$,则 $\textbf{TIME}(f(n)) \subsetneq \textbf{TIME}(g(n))$。

Proof.(对角线方法) 我们来证明一个否定命题 $\textbf{TIME}(g(n)) \not\subseteq \textbf{TIME}(f(n))$。

构造反例L: 输入 x,在$\mathbb{M}_x$计算$g(|x|)$ 步;如果完成模拟,输出 $\mathbb{M}_x(x)$ 的反转,否则输出 0。由定义,$L \in \textbf{TIME}(g(n))$,且$\mathbb{D}$在时间$g(n)$判定$L$。

接下来证明不存在图灵机$\mathbb{M}$,能够在$f(n)$时间内判定L。

假定 $L \in \textbf{TIME}(f(n))$,且有$M_z(z)$能判定。

我们取充分大的 z,由 1.3.2,我们知道 $g(|z|) \gg f(|z|)\log(|z|)$,$\mathbb{D}$ 可以模拟完运算。因为两者都判定L,s$\mathbb{D}(z) = \mathbb{M}_z(z)$;同时,按照输出结果我们有 $\mathbb{D}(z) = \overline{\mathbb{M}_z(z)}$,这便是一个矛盾。

□

解读: 1.由于g时间可构造,所以$\mathbb{D}$可以用一个$\mathbb{T}_{g(n)}$的时钟图灵机掐表来实现.也就是说,

$$\mathbb{D} \longleftarrow \mathbb{T}_{g(n)} \wedge \lambda z.\mathbb{U}(z, z)$$

,$\wedge$表示掐表.

2.证明过程中的反证法看起来不那么显然,但是是基于一个直观的命题上的:

$$\textbf{TIME}(f(n)) \subset \textbf{TIME}(g(n))$$

3.我们**为什么**可以取这样的 $L \in \textbf{TIME}(f(n))$,$M_z(z)$ 判定之,并且这里的z还能充分大?其实,我们是这么做的:取这样的z,s.t.$\mathbb{M}_z$判定L的时间函数$T(n) \leqslant Cf(n)$.

4.对角线方法的图例:

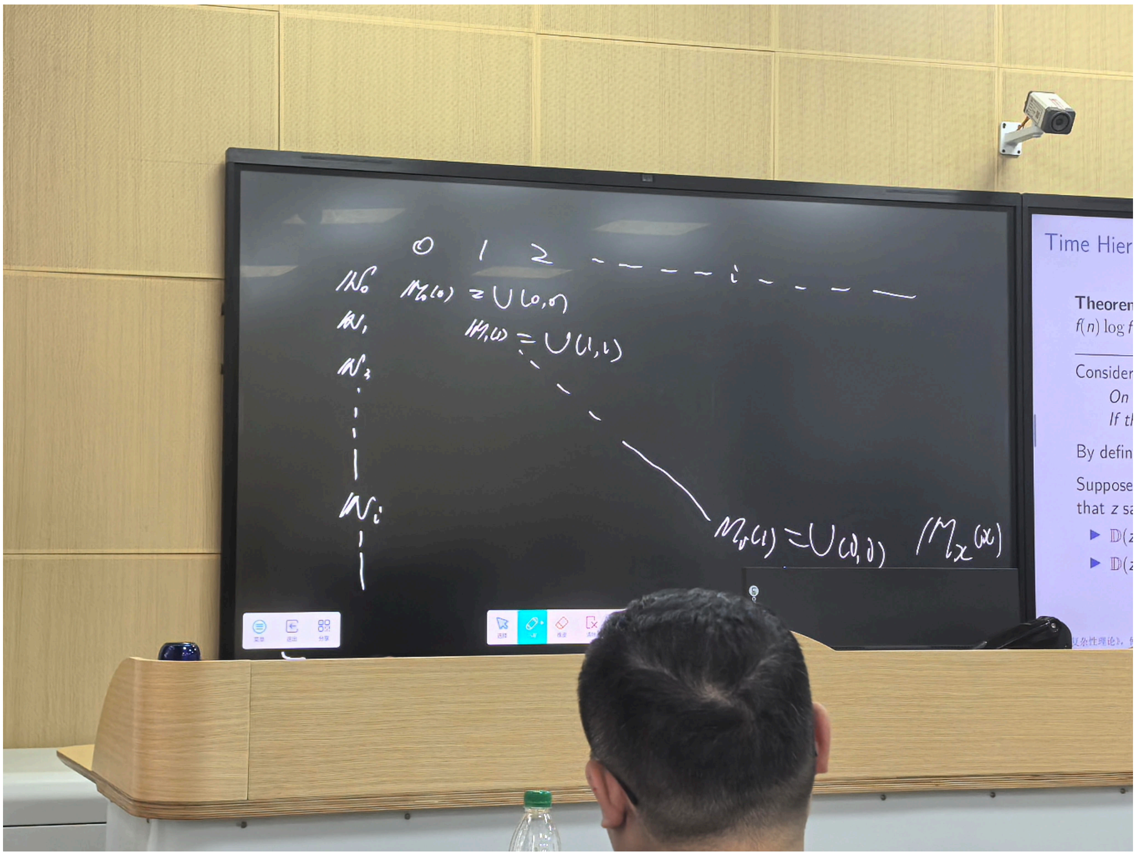

Figure 2: 对角线的来源是用通用图灵机模拟一个图灵机的时候,在对角线的时候取反.

初等函数类的定义是如下的:

$$\mathrm{EXP} = \bigcup_{c>1} \mathrm{TIME}(2^{n^c})$$

$$2\mathrm{EXP} = \bigcup_{c>1} \mathrm{TIME}\left(2^{2^{n^c}}\right)$$

$$3\mathrm{EXP} = \bigcup_{c>1} \mathrm{TIME}\left(2^{2^{2^{n^c}}}\right)$$

$$\vdots$$

$$\mathrm{ELEMENTARY} = \mathrm{EXP} \cup 2\mathrm{EXP} \cup 3\mathrm{EXP} \cup \ldots$$

非确定图灵机的谱系问题也有对应的定理和证明方法.这里的定理稍显奇怪(因为是$f(n+1)$).

Theorem 1.9.2 (时间谱系定理)**.**
若函数 f, g 是时间可构造的,而且 $f(n+1) = o(g(n))$,则 $\mathbf{NTIME}(f(n)) \subsetneq \mathbf{NTIME}(g(n))$。

Proof. 我们来设计一个精致的图灵机$\mathbb{Z}$来证明这个命题.

1. 首先,$\mathbb{Z}$的输入读写头和第一条工作带读写头往右扫描.
   - 若输入不形如$1^n \wedge n > 1$,直接停机并输出 0.
   - **第一条工作带**上第一个位置写 1,然后随机写 0 和 1.我们记$\{h_i\}$为工作带上写 1 的位置序列.
2. **第二条工作带**上,$\mathbb{Z}$直接模拟那些所有非确定图灵机和$\mathbb{T}_{2f(n)}$的硬连接.不妨设这些图灵机为$\{\mathbb{L}_i\}$.$\mathbb{Z}$模拟完$\{\mathbb{L}_i\}$之后,需要做如下操作来接着模拟$\{\mathbb{L}_{i+1}\}$:
   - 暂停第 1 步的操作;

- 在**第三条工作带**上构造$1^{h_{i-1}+1}$.这是因为我们假定第三条工作带上保留着$1^{h_{i-2}+1}$,我们只需要再写$h_{i-1}-h_{i-2}$个 1.这里额外开销的时间是$O(h_i)$是线性的;
- 把$\mathbb{L}_{i-1}$的编码复制到第三条工作带上.这也是线性时间的;
- 恢复第一步时候的读写头;
- 恢复输入带和第一条工作带读写头的同步全速向右扫描。与此同时,$\mathbb{Z}$用暴力法计算$\mathbb{L}_{i-1}\left(1^{h_{i-1}+1}\right)$.计算完后,$\mathbb{Z}$将结果写在第二条工作带上, 与此同时, 在第一条工作带上写 1,这个写了 1 的格子的地址就是h_i.

3. 当输入带结束扫描之后,做如下操作:
   - 若$n=h_i$,$\mathbb{Z}$接受$1^n \iff \mathbb{L}_{i-1}\left(1^{h_{i-1}+1}\right)=0$;
   - 否则, $\mathbb{Z}$非确定的让$\mathbb{V}(\mathbb{L}_{i-1},1^{n+1})$计算$g(n)$步.

设L是$\mathbb{Z}$接受的语言,来考察一下时间复杂度.关键性的是$\mathbb{V}(\mathbb{L}_{i-1},1^{n+1})$的计算开销,不难看出其余步骤全是线性的;所以$L\in \mathbf{TIME}(g(n))$.

另一方面,用**反证法**证明$L\notin \mathbf{TIME}(f(n))$. 假设 $L\in \mathbf{NTIME}(f(n))$, 且有$\mathbb{N}_i$能判定.由线性加速定理,我们假定时间函数是$2f(n)$.根据定义,$\mathbb{L}_i$接受$L$,那么当$i$充分大的时候,我们有

$$
\begin{aligned}
\mathbb{L}_i\left(1^{h_i+1}\right) &= \mathbb{Z}\left(1^{h_i+1}\right)\\
&= \mathbb{L}_i\left(1^{h_i+2}\right)\\
&= \mathbb{Z}\left(1^{h_i+2}\right)\\
&= \cdots\\
&= \mathbb{L}_i\left(1^{h_{i+1}}\right)\\
&= \mathbb{Z}\left(1^{h_{i+1}}\right)\\
&\neq \mathbb{L}_i\left(1^{h_i+1}\right)
\end{aligned}
$$

这一矛盾说明结论成立.

□

作为本章的结束, 来看一个定理。

Theorem 1.9.3 (间隙定理).
设 $r(x)\geq x$ 为可计算全函数（不一定时间可构造的）。存在可计算全函数 $b(x)$, $\mathrm{TIME}(b(x))=\mathrm{TIME}(r(b(x)))$。

Proof. 记 $k_0<k_1<\cdots$, $k_0=0$, $k_{i+1}=r(k_i)+1$。并记 $P(i,k)$ 为“$\forall j\leq i$, $|z|=i$, $M_i(z)$ 要么 k 步内停机, 要么 $r(k)$ 步不停机”。我们设 $n_i=\sum_{j=0}^{i}|\Gamma|^i$ 为图灵机 M_i 的符号串编码长度为 i 的总数, 讲这些符号串翻过去作为输入（也就是对角线方法）。

计算结果肯定不超过 n_i 个, 那么由抽屉原理, 存在 $j\leq n_i$, $P(i,k_j)=1$。定义 $b(i)=k_j$, 所以对所有 i, $P(i,b(i))=1$。

取 $L\in\mathrm{TIME}(r(b(x)))$。对足够大的 x, 总存在一个图灵机 M, 它要在 $b(|x|)$ 步停机, 要么 $r(b(x))$ 步内不停机, 后者显然不符合, 这就说明了 $L\in\mathrm{TIME}(b(x))$。

□