B0911005Y-01: Introduction to Theory of Computation **2023 Spring**

Homework 8 — June 9

Lecturer: Mingji Xia *Completed by:* 吉骏雄

第 8.1 次作业: 9.4, 9.5, 8.4, 8.6, 8.25. 选做: 9.12. 注: 书图 9-5 x_2 与其右下方的非门之间少连了一根导线.

9.4 如在图 8.1中所做的那样, 标出图 8.2所描绘的电路中所有门计算出的值, 从而说明它在输入 0110 上的计算历史.

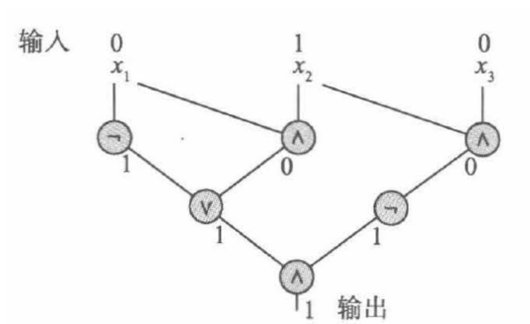

图 8.1. 9.4 题图 1

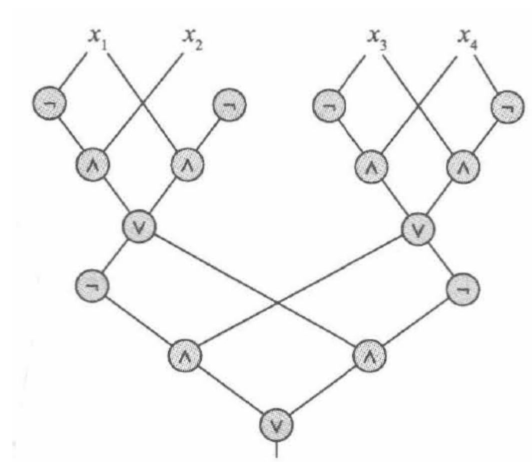

图 8.2. 9.4 题图 2

解 结果如图 8.3所示.

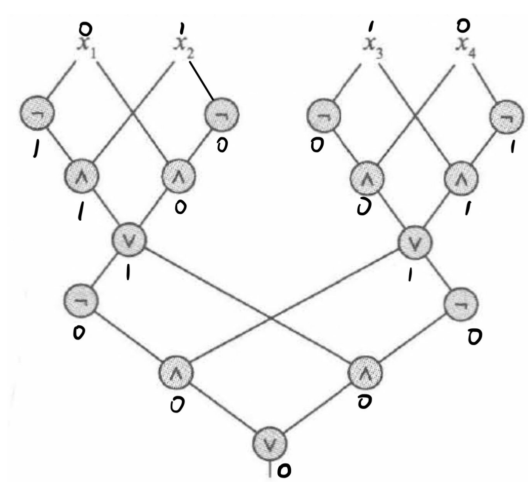

图 8.3. 9.4 题答案

9.5 给出三个输入变量上计算奇偶函数的电路, 并说明它在输入 011 上的计算历史.

解 结果如图 8.4所示.

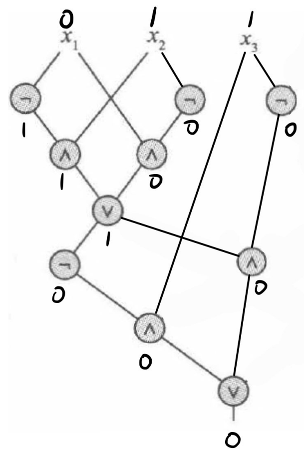

图 8.4. 9.5 题答案

8.4 证明 PSPACE 在并、补和星号运算下封闭.

证明 **并运算:**

$\forall L_1, L_2 \in$ PSPACE, 根据定义, 存在两个图灵机 M_1, M_2 分别能够判定 L_1, L_2, 且所需空间 $f_1(n), f_2(n)$ 均为多项式空间. 我们构造一个图灵机 M': M' = "对于输入 ω, 其中 ω 是字符串:

1. 将输入复制一份在带上输入右面, 左面的原输入暂时封存, 用不会出现在 M_1, M_2 带字符表的符号将之分隔开.
2. 使用右侧复制的内容, 模拟在输入 ω 上运行 M_1.
3. 如果 M_1 接受, 则接受; 如果 M_1 拒绝, 则清空分隔符右侧的内容, 返回原输入状态.
4. 模拟在输入 ω 上运行 M_2.
5. 如果 M_2 接受, 则接受; 如果 M_2 拒绝, 则拒绝.

"

可以看到, M' 可能改写的格子数最多为 $\max[f_1(|\omega|), f_2(|\omega|) + |\omega| + 1]$, 复杂多也是多项式级别的. M' 能判定 $L_1 \cup L_2$, 因此 $L_1 \cup L_2 \in$ PSPACE. 因此 PSPACE 在并运算下封闭.

星号运算:

设 M 能在 $O(n^a)$ 空间内判定 L. 我们构造一个图灵机 M': M' = 对于输入 ω, 其中 $\omega = \omega_1\omega_2\omega_3 \dots \omega_n$ 是字符串:

1. 若 $\omega = \varepsilon$, 则接受.
2. 非确定地选择 $m, 1 \le m \le n$. 非确定地将 ω 分割为 $s_1 s_2 s_3 \dots s_m$, 其中每个 s_i 都非空.

3. 对所有 $k = 1 \sim m$:
4. 在 M 上运行 s_k, 如果 M 拒绝, 则拒绝;
5. 如果 M 接受, 则接受.

可以看到, M' 最多需要 $O(n)$ 的空间保存划分结果, 之后的运算都是多项式空间内可完成的. 因此 M' 能在多项式空间内判定 L^*, 因此 $L^* \in$ PSPACE. 因此 PSPACE 在星号运算下封闭.

补运算:

$\forall L \in$ PSPACE, 根据定义, 存在图灵机 M 能够判定 L, 且所需空间 f 为多项式空间. 我们构造一个图灵机 M', 其与 M 的区别仅为接受状态与拒绝状态互换, 即 q_{accept} 与 q_{reject} 互换. 这样构造的图灵机依然可以对任意输入停机, 所需的计算空间一致, 且接受的语言恰为 $\overline{L}$. 因此 PSPACE 在补运算下封闭. □

8.6 证明 PSPACE 难的语言也是 NP 难的.

证明 考虑两个特殊的语言: TQBF 以及 SAT, 以及分别识别这两个语言的两台图灵机 M_1, M_2. 已经知道前者是一个 PSPACE 完全的语言, 后者是一个 NP 完全的语言. 我们证明 TQBF 是 NP 难的, 可以间接地证明 SAT $\leq_p$ TQBF, 存在一个多项式时间的图灵机 M 能够通过将 SAT 多项式时间规约到 TQBF 的方法识别它: $M =$ "对于输入 ϕ, 其中 ϕ 是一个布尔表达式:

1. 为输入的布尔表达式外侧添加一对括号.
2. 检查输入的布尔表达式 ϕ 中所有的变量, 为每个变量 x 在表达式前端添加一个对应的存在量词 $\exists x$. 这样得到一个前束范式形式的全量词化的布尔表达式 ψ, 输出之.

" 接下来模拟在输入 ψ 上运行 M_2 (能识别 TQBF 的图灵机), 其接受与拒绝同 M 的输入 "是否在 SAT 中" 的结果一致.

由于除添加括号外, 操作次数等于变量数, 故该规约是多项式时间内的; 即可计算函数 $f : \phi \rightarrowtail \psi$ 是多项式时间可计算函数. 因此得到 SAT $\leq_p$ TQBF.

由于 SAT 是 NP 完全的, 因此 $\forall L \in$ NP-hard, $\forall L' \in$ PSPACE-hard, 均有 $L \leq_p$ SAT $\leq_p$ TQBF $\leq_p L' \Longrightarrow L \leq_p L'$, 故 PSPACE 难的语言也是 NP 难的. □

8.25 梯子 (ladder) 是一个字符串的序列 $s_1, s_2, \ldots, s_k$, 其中每个字符串与前一个字符串恰好只在一个字母上不同. 例如, 下面是一个英文单词的梯子:

head, hear, near, fear, bear, beer, deer, deed, feed, feet, fret, free

令 $\text{LADDER}_{\text{DFA}} = \{\langle M, s, t\rangle \mid M$ 是一个DFA, $L(M)$ 包含一个以字符串s 开头、以字符串t 结束的梯子$\}$. 证明 $\text{LADDER}_{\text{DFA}}$ 属于 PSPACE.

证明 根据 Savitch 定理, 我们有:

$$\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$$

由于多项式的平方依旧是多项式, 所以:

$$\text{NPSPACE} = \bigcup_{i=0}^{\infty} \text{NSPACE}(n^i) \subseteq \bigcup_{i=0}^{\infty} \text{SPACE}(n^{2i}) = \text{PSPACE}$$

所以我们可以通过证明 $\text{LADDER}_{\text{DFA}} \in \text{NPSPACE}$, 来证明题目中的 $\text{LADDER}_{\text{DFA}} \in \text{PSPACE}$. 构造一个非确定型图灵机 M, 其能够在多项式空间内判定 $\text{LADDER}_{\text{DFA}}$. $M =$ "对于输入 $\langle M, s, t\rangle$, 其中 M 是一个 DFA, s, t 是字符串:

1. 在原输入的最末端添加一个数字 $t = |\Sigma|^{|s|}$, 这表示了和 s 长度相同的字符串的总可能个数.
2. 从 s 开始, 按照 M 的转移函数, 模拟 M 在输入 s 上的运行. 如果 M 在 s 上拒绝, 则拒绝.
3. 如果 s 与 t 相同, 则接受.
4. 非确定地选择 s 中的任意一个变量, 并非确定地 (即同时) 将其替换为 Σ 中的任意一个字母 (除了原本的字母以外), 并将其作为新的字符串, 覆写在原本 s 的位置上.
5. 对 t 的值减去 1.
6. 如果 $t = 0$, 说明已经遍历过了所有可能性, 依旧没有抵达可以被接受的 t, 则拒绝.
7. 返回第 2 个步骤继续执行.

"

可以发现, 空间使用不超过 $f(n) = n + |s| + c + |s|\log|\Sigma|$, 其中 n 是输入字符串长度, c 是分隔末尾添加数字所用的分隔符长度. 显然空间使用量是多项式的, 宽松估计也有 $f(n) = \text{O}(n^2)$. 因此 $\text{LADDER}_{\text{DFA}} \in \text{NPSPACE}$, 从而 $\text{LADDER}_{\text{DFA}} \in \text{PSPACE}$. $\square$

第 8.2 次作业: 8.3, 8.1, 8.11(a), 8.33

8.3 考虑下图所示的广义地理学游戏, 其中起始结点就是由无源箭头指向的结点. 选手 I 有必胜策略吗？选手 II 呢？给出理由.

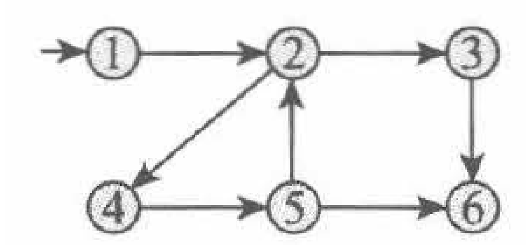

图 8.5. 8.3 题图

解 选手 II 有必胜策略, 选手 I 没有.

选手 I 第一步必须要从结点 1 前往结点 2. 接下来, 选手 II 可以选择从结点 2 前往结点 4. 之后, 选手 I 必须要从结点 4 前往结点 5. 这时, 选手 II 只需选择从结点 5 前往结点 6, 即可获得胜利, 因为结点 6 出度为 0, 选手 I 无路可走.

8.1 证明对于任意函数 $f : \mathbb{N} \to \mathbb{R}^+$, 其中 $f(n) \geq n$, 不论用单带图灵机模型还是用双带只读输入图灵机模型, 所定义的空间复杂性类 SPACE($f(n)$) 总是相同的.

解 需要证明两个命题: 单带图灵机模型定义的 SPACE($f(n)$) 一定是双带只读输入图灵机模型定义的 SPACE($f(n)$) 的子集, 反之亦然. 分别记录两种定义下的集合为 S_n 和 D_n, 则需要证明 $S_n \subseteq D_n$ 且 $D_n \subseteq S_n$.

$S_n \subseteq D_n$**:**

$\forall L \in S_n$, 根据定义, 存在单带图灵机 M 能够判定 L, 且所需空间 f 为多项式空间. 我们构造一个图灵机 M', 其与 M 的区别为: 将输入带改为只读输入带, 并在运行开始之后立即将只读输入带上的内容全部转写到读写工作带上; 之后 M' 完全在读写工作带上运行, 规则同 M. 这样构造的图灵机依然可以对任意输入停机, 所需的计算空间完全一致, 且接受的语言恰为 L. 因此 $S_n \subseteq D_n$.

$D_n \subseteq S_n$**:**

$\forall L \in D_n$, 根据定义, 存在双带只读输入图灵机 M 能够判定 L, 且所需空间 f 为多项式空间. 我们构造一个图灵机 M', 其与 M 的区别为: 将只读输入带和读写工作带合并成一条纸带, 只读输入只占有限长, 不变, 原读写工作带区域置于纸带上原只读输入带区域之后; 之后 M' 运行, 规则同 M, 并以插入特殊字符的形式记录原两个读写头在原两条纸带上的位置. 这样构造的图灵机依然可以对任意输入停机, 所需的计算空间为 $n + f(n) + \mathrm{O}(1) = \mathrm{O}(f(n))$ (比原本的计算空间多了一个除标记读写头外不可修改的输入字符串, 两带区域的分隔符, 以及标记原读写头位置的字符), 且接受的语言恰为 L. 因此 $D_n \subseteq S_n$.

两个集合相互包含, 因此 $S_n = D_n$, 即不论用单带图灵机模型还是用双带只读输入图灵机模型, 所定义的空间复杂性类 SPACE($f(n)$) 总是相同的.

8.11(a) 令 ADD $= \{\langle x, y, z\rangle \mid x, y, z > 0$ 且为二进制整数, $x + y = z\}$, 证明 ADD $\in L$.

证明 我们构造一个双带只读输入图灵机 M, 其能够在对数空间内判定 ADD. M = "对于输入 $\langle x, y, z\rangle$, 其中 x, y, z 是二进制整数且均 > 0:

1. 仅用两位二进制数和一个整数进行读写工作带上的计算, 即带上始终只有两位与一个整数, 分别是进位 c, 和位 s, 位数 i. 令 $i = 0$
2. 考虑 x, y 的右数第 i 位 x_i, y_i ($i = 0$ 代表的是最低位), 逐位将 $x_i + y_i + c$ 记录到带上的 $\overline{cs}$ 两 bit 位置处, 即求和后的 2 位二进制数替代了原本的数据.
3. 如果 $s \neq z_i$, 则拒绝. (以上, 若 x, y, z 中某一个数的位数不足, 则补充 0).
4. 如果 $i \neq n$, 其中 n 为 x, y, z 三者最高位位数的最大值, 则将 i 加 1, 回到第 2 步.
5. 如果 $c = 0$, 则接受. 否则, 拒绝.

"

可以发现, 空间使用不超过 $f(n) = 2 + \log(n)$. 因此 ADD $\in L$. □

8.33 设 A 是由正确嵌套的圆括号组成的语言. 例如, (()) 和 (()(())) () 属于 A. 而)(则不属于 A. 证明 A 属于 L.

证明 我们构造一个双带只读输入图灵机 M, 其能够在对数空间内判定 ADD. M = "对于输入 ω, 其中 ω 是圆括号组成的字符串:

1. 仅用一个整数进行读写工作带上的计算, 即带上始终只有一个整数 i. 令 $i = 0$.
2. 从左向右读取 ω 的每一个字符, 如果是左括号, 则将 i 加 1; 如果是右括号, 则将 i 减 1; 进入下一步的判断.
3. 如果 $i < 0$, 则拒绝. 否则回到第二步继续读取下一个字符, 并进行相关操作.
4. 读完所有字符之后, 如果 $i = 0$, 则接受. 否则, 拒绝.

"

可以发现, 空间使用不超过 $f(n) = \log(n) + \mathrm{O}(1)$. 因此 ADD $\in L$. □

第 8.3 次作业：8.16，8.18，9.9，9.18

8.16 回忆一下，在有向图中，如果每一对结点间都有双向的有向路径连接，则它称为强连通的 (strongly connected)。令 STRONGLY-CONNECTED = $\{\langle G\rangle \mid G$ 是强连通图$\}$. 证明 STRONGLY-CONNECTED 是 NL 完全的。

证明 我们构造一个双带只读输入图灵机: M = "对于输入 $\langle G\rangle$, 其中 G 是一个有向图:

1. 向工作带上写入两个数字 n, b, e, t, s, 分别表示 G 的结点数, 以及当前正在遍历的结点编号 (编号不超过 n). 令 $b = e = 1$.
2. 从结点 $b = 1$ 开始, 遍历所有结点. 对于每一个结点, e 又遍历所有结点. 对每一组结点对, 重复以下步骤:
3. 如果 $b = e$, 则跳过此次循环, 并回到第 2 步.
4. 令 $t = b$, $s = 0$.
5. 非确定地选择一条以 t 为起点的边, 将 t 改为该边的终点. 如果不存在这样的边, 则拒绝. 如果 t 已经是 e, 则回到第 2 步.
6. s 加 1. 如果 $s < n - 1$, 则回到第 5 步. 否则, 所有可能简单路径都已经被某个分支尝试过, 则拒绝.
7. 如果之前没被拒绝, 最终接受.

"

可以发现, 空间使用不超过 $f(n) = 5\log(n) + \mathrm{O}(1)$. 因此 STRONGLY-CONNECTED $\in$ NL.

构造一个对数空间转换器 M, 它能将 PATH 映射到 STRONGLY-CONNECTED: M = "对于输入 $\langle G, s, t\rangle$, 其中 G 是一个有向图:

1. 将 G 写至输出带上.
2. 对于每一个结点 v, 在输出带上添加一条从 v 指向 s 的边, 和一条从 t 指向 v 的边.

"

只要 G 中有从 s 到 t 的路径, 那么 $f(M)$ 就能被 STRONGLY-CONNECTED 接受. 因此 PATH $\leq_L$ STRONGLY-CONNECTED.

因此, STRONGLY-CONNECTED 是 NL 完全的. □

8.18 证明 $\mathrm{A}_{\mathrm{NFA}}$ 是 NL 完全的。

证明 我们构造一个双带只读输入图灵机: M = "对于输入 $\langle N, \omega\rangle$, 其中 N 是一个 NFA, ω 是一个字符串:

1. 约定 ω 从左向右数的的第 i 个字符为 ω_i.
2. 选取初始状态 q_0 开始. 记录 $i = 0$ 于带上.

3. 从 $i \in \{1, 2, \ldots, n\}$ 中, 非确定地选取一个状态 q_i, 检查是否有 $\delta(q_{i-1}, q_i) = \omega_i$ (注意, 此时空转移也算入其中). 如果不是, 则拒绝.

4. 检查 q_n 是否为结束状态. 如果是, 则接受; 否则, 拒绝.

”

可以发现, 空间使用不超过 $f(n) = \log(n) + \mathrm{O}(1)$. 因此 $A_{\mathrm{NFA}} \in \mathrm{NL}$.

构造一个对数空间转换器 M, 它能将 PATH 映射到 $\mathrm{A_{NFA}}$: $M =$ "对于输入 $\langle G, s, t \rangle$, 其中 G 是一个有向图:

1. 将 G 中的每一个点写作一个状态, 写至输出带上.

2. 将 G 中的每一条 (有向) 边写作一个转移函数, 写至输出带上, 其转移条件为任意一个字符.

3. 将 $|V|-1$ 个 1 作为输入 ω, 写至输出带上. 其中 $|V|$ 为 G 的结点数.

”

只要 G 中有从 s 到 t 的路径, 那么 $f(M)$ 就能被 $\mathrm{A_{NFA}}$ 接受, 因为输入能让 NFA 接受. 因此 PATH $\leq_L$ $\mathrm{A_{NFA}}$.

因此, $\mathrm{A_{NFA}}$ 是 NL 完全的. □

9.9 证明若 $\mathrm{NP} = \mathrm{P^{SAT}}$, 则 $\mathrm{NP} = \mathrm{coNP}$

证明 假设 $\mathrm{NP} = \mathrm{P^{SAT}}$.

$\forall L \in \mathrm{P^{SAT}}$, 根据定义, 存在采用喻示 SAT 的确定型喻示图灵机 M 能够判定 L. 将 M 的接受状态与拒绝状态互换, 即 q_{accept} 与 q_{reject} 互换, 得到的图灵机 M'; 由于 M' 是采用喻示 SAT 的确定型喻示图灵机, 这样构造的图灵机依然可以对任意输入停机, 且所需的计算时间一致. M' 接受的语言恰为 $\overline{L}$, 所以 $\overline{L} \in \mathrm{P^{SAT}}$.

因为 $\forall L \in \mathrm{coNP}$, 均有 $\overline{L} \in \mathrm{NP}$, 所以 $\overline{L} \in \mathrm{NP} = P^{\mathrm{SAT}} \Longrightarrow L \in \mathrm{P^{SAT}}$. 因此 $\mathrm{coNP} \subseteq \mathrm{P^{SAT}}$.

因为 $\forall L \in \mathrm{P^{SAT}}$, 均有 $\overline{L} \in \mathrm{P^{SAT}} = \mathrm{NP}$, 所以 $L \in \mathrm{coNP}$. 因此 $\mathrm{P^{SAT}} \subseteq \mathrm{coNP}$.

综上, $\mathrm{P^{SAT}} = \mathrm{coNP}$. □

9.18 定义唯一满足 (unique-sat) 问题为：USAT $= \{\langle \phi \rangle \mid \phi$ 是一个只有唯一满足赋值的布尔公式$\}$。证明 $\mathrm{USAT} \in \mathrm{P^{SAT}}$.

证明 我们构造一个非确定型图灵机 M, 其能够在多项式时间内判定 $\overline{\mathrm{USAT}}$. $M =$ "对于输入 ϕ, 其中 ϕ 是一个布尔公式:

1. 先仿照 SAT 运行, 检查 ϕ 是否可满足. 如果不能, 则接受.

2. 非确定地选择两组赋值, 检查 ϕ 在这两组赋值下是否都可满足. 如果是, 则接受. 如果否, 则拒绝.

”

如果 ϕ 不是唯一可满足的, 即 $\phi \in \overline{\mathrm{USAT}}$, 那么 M 一定是两种情况之一: 不可满足的 (在第一步中被接受), 或者有两个不同的可满足赋值 (在第二步中被接受). 因此 M 能够判定 $\overline{\mathrm{USAT}}$, 且只需要多项式时间 ($\mathrm{SAT} \in \mathrm{NP}$), 所以 $\mathrm{USAT} \in \mathrm{coNP} \subseteq \mathrm{P^{SAT}}$, 证毕. □

第 8.4 次作业（书上均有答案，不需要提交）：8.7，9.1-9.3