有限状态自动机及其应用

徐哲安

杭州学军中学

February 3, 2021

引入

有限状态自动机是一类基础的计算模型，在现实生活中有广泛的应用。同样，有限状态自动机也能协助我们解决各类信息学竞赛中的问题，具有广泛的应用前景。

引入

有限状态自动机是一类基础的计算模型，在现实生活中有广泛的应用。同样，有限状态自动机也能协助我们解决各类信息学竞赛中的问题，具有广泛的应用前景。

本次交流将会对有限状态自动机进行探究，并将相关的理论加以梳理，总结有限状态自动机在信息学竞赛中的应用。

确定性有限状态自动机

Definition

确定性有限状态自动机（*Deterministic Finite Automaton, DFA*）是一个五元组 $(Q, \Sigma, \delta, q_0, F)$，其中

- ▷ Q 是一个有限**状态集合**；
- ▷ Σ 是一个有限**字符集**；
- ▷ $\delta : Q \times \Sigma \to Q$ 是**转移函数**；
- ▷ $q_0 \in Q$ 是**开始状态**；
- ▷ $F \subset Q$ 是**接受状态集合**。

状态图可以直观地描述一个 DFA。

DFA 的计算

DFA 的**计算**：从开始状态出发，读入一个字符后沿着对应转移边达到新的状态。若完整读入串 w 后，所处的状态为接受状态，则称该自动机**接受** w。

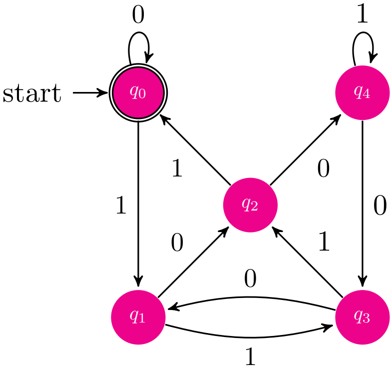
DFA 的计算

DFA 的**计算**：从开始状态出发，读入一个字符后沿着对应转移边达到新的状态。若完整读入串 w 后，所处的状态为接受状态，则称该自动机**接受** w。

形式语言（简称**语言**）是任意 Σ 上串的集合。所有被 DFA M 接受串组成了语言 $L(M)$，称 M **识别** $L(M)$。如果一个语言能被某个 DFA 识别，则称它为**正则语言**（*Regular Language*）。

一个 DFA 的例子

设计一个 DFA 能识别 5 的倍数的二进制串。



非确定性有限状态自动机

非确定性有限状态自动机（*Nondeterministic Finite Automaton*, NFA）在 DFA 的基础上加入 ϵ 转移边，同时任何一对点和转移字符可能存在多个后继。类似可以给出 NFA 计算的定义，需要注意的是，只要存在一条路径能到达接受状态，就称 NFA 接受这个串。

非确定性有限状态自动机

非确定性有限状态自动机（*Nondeterministic Finite Automaton*, NFA）在 DFA 的基础上加入 ϵ 转移边，同时任何一对点和转移字符可能存在多个后继。类似可以给出 NFA 计算的定义，需要注意的是，只要存在一条路径能到达接受状态，就称 NFA 接受这个串。

似乎一个可能的结论是 NFA 比 DFA 能识别更多的语言类?

非确定性有限状态自动机

非确定性有限状态自动机（*Nondeterministic Finite Automaton*, NFA）在 DFA 的基础上加入 ϵ 转移边，同时任何一对点和转移字符可能存在多个后继。类似可以给出 NFA 计算的定义，需要注意的是，只要存在一条路径能到达接受状态，就称 NFA 接受这个串。

似乎一个可能的结论是 NFA 比 DFA 能识别更多的语言类?

Theorem

每一个 *NFA* 都等价于某一个 *DFA*，反之亦然。两个机器等价，即它们能识别的语言类相同。

NFA 与 DFA 的对比

使用**幂集构造**（*Powerset construction*）即可得到 NFA 对应等价的 DFA。这也说明了对于同一正则语言，描述它的 NFA 可能远小于 DFA，这也是 NFA 的优势所在。

NFA 与 DFA 的对比

使用**幂集构造**（*Powerset construction*）即可得到 NFA 对应等价的 DFA。这也说明了对于同一正则语言，描述它的 NFA 可能远小于 DFA，这也是 NFA 的优势所在。

下面是一个经典的例子，使用正则表达式描述即为

$$(a+b)^*a(a+b)(a+b)\cdots(a+b)$$

这是一个长度为 $\Theta(n)$ 的正则表达式，意义是全体满足倒数第 $n+1$ 个字符为 a 的 ab 串。我们可以构造出一个大小为 $\Theta(n)$ 的 NFA，但是其最小 DFA 却是 $\Theta(2^n)$ 的。

DFA 与 NFA 的计算

DFA 计算一个串的时间复杂度为 $\mathcal{O}(n)$，而 NFA 计算的时间复杂度为 $\mathcal{O}(ns^2)$。通过 bitset 维护转移边集合，NFA 的计算可以做到 $\mathcal{O}(\frac{ns^2}{\omega})$ 的时间复杂度。

DFA 与 NFA 的计算

DFA 计算一个串的时间复杂度为 $\mathcal{O}(n)$，而 NFA 计算的时间复杂度为 $\mathcal{O}(ns^2)$。通过 bitset 维护转移边集合，NFA 的计算可以做到 $\mathcal{O}(\frac{ns^2}{\omega})$ 的时间复杂度。

进一步优化需要使用 **Method of Four Russians**。也就是按照 $\Theta(\log n)$ 的大小分块，预处理每个块中子集的转移边集合，可以做到 $\mathcal{O}(\frac{ns^2}{\omega \cdot \log n})$ 的时间复杂度。

正则表达式

Definition

对于一个**正则表达式**（*Regular Expression*）R，称 $L(R)$ 为正则表达式 R 对应的形式语言。正则表达式 R 可以是

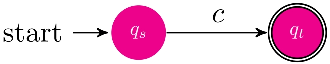
1. c $(c \in \Sigma)$，表示语言 $L(R) = \{c\}$；
2. ϵ，表示语言 $L(R) = \{\epsilon\}$；
3. $\emptyset$，表示语言 $L(R)$ 为空语言；
4. $(R_1 + R_2)$，表示语言 $L(R) = L(R_1) \cup L(R_2)$；
5. $(R_1 R_2)$，表示语言 $L(R) = \{uv \mid u \in L(R_1), v \in L(R_2)\}$；
6. (R_1^*)，表示语言 $L(R) = \{u_1 u_2 \cdots u_n \mid u_i \in R_1, n \in \mathbb{N}\} \cup \{\epsilon\}$。

正则表达式与 FSM 所能表述的语言类是相同的。

正则表达式转 NFA

我们可以使用 **Thompson 构造法**（*Thompson's construction*），根据正则表达式 R 的构成，有如下五种情况：

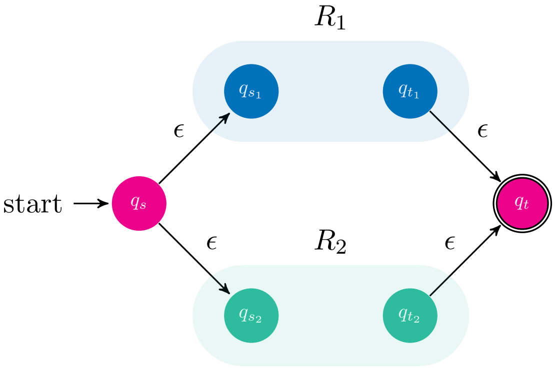
若 $R = c \ (c \in \Sigma \cup \{\epsilon\})$，



若 $R = \emptyset$，

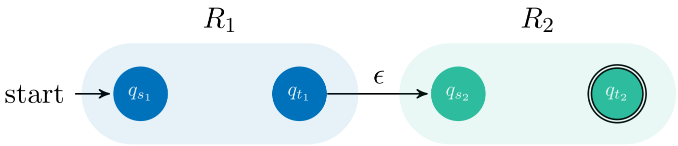
正则表达式转 NFA

若 $R = (R_1 + R_2)$，



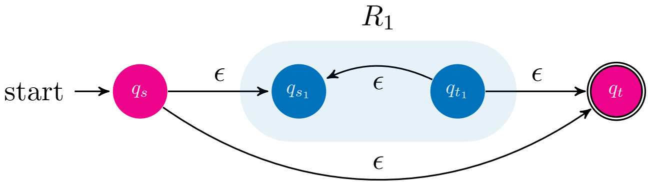
正则表达式转 NFA

若 $R = (R_1 R_2)$，



正则表达式转 NFA

若 $R = (R_1^*)$，



正则表达式与正则语言

同样 DFA 也能转化为正则表达式，具体构造可以参见论文。

正则表达式与正则语言

同样 DFA 也能转化为正则表达式，具体构造可以参见论文。

我们将正则表达式与正则语言建立了联系，这就使得我们可以从其它视角，考虑正则语言的更多性质。

论文中提供了一种判断两个含变量正则表达式是否相同的方法，能帮助我们发现正则语言的其它运算定律；介绍了正则语言对于一些运算的封闭性，这可以作为构造复杂 FSM 的工具；最后介绍了正则语言的泵引理，可以用于证明一个语言的非正则性。

DFA 的等价类与最小化

Definition

对于 *DFA* 的两个状态 p, q，我们定义关系 $p \sim q$ 当且仅当：对于任意输入串 $w = w_1 w_2 \cdots w_k$ $(k \geq 0)$，$\hat{\delta}(p, w) = \delta(\cdots\delta(\delta(p, w_1), w_2) \cdots, w_k)$ 是接受状态当且仅当 $\hat{\delta}(q, w)$ 是接受状态。

等价关系 $\sim$ 将 DFA 的状态划分为若干个等价类。剔除初始状态无法到达的状态后，我们可以将同一个等价类缩成一个点，得到最小化的 DFA。

那么，如何找到一个 DFA A 所有的等价类？

等价类划分算法

等价类划分算法基于不断对等价类进行划分。定义 $p \sim_k q$ 表示对于任意长度 $\leq k$ 的串 w，均有 $\hat{\delta}(p, w)$ 是接受状态当且仅当 $\hat{\delta}(q, w)$ 是接受状态。$\sim_k$ 是一个等价关系，DFA A 被划分成了等价类集合 Π_k。显然 $\Pi_0 = \{Q \backslash F, F\}$，容易从 Π_k 推出 Π_{k+1}。

等价类划分算法

等价类划分算法基于不断对等价类进行划分。定义 $p \sim_k q$ 表示对于任意长度 $\leq k$ 的串 w，均有 $\hat{\delta}(p, w)$ 是接受状态当且仅当 $\hat{\delta}(q, w)$ 是接受状态。$\sim_k$ 是一个等价关系，DFA A 被划分成了等价类集合 Π_k。显然 $\Pi_0 = \{Q \backslash F, F\}$，容易从 Π_k 推出 Π_{k+1}。

若依次求出了等价类集合 $\Pi_0, \Pi_1, \cdots, \Pi_m$ 且 $\Pi_{m-1} = \Pi_m$。必然有 $\Pi_{m-1} = \Pi_m = \Pi_{m+1} = \Pi_{m+2} = \cdots$，那么 Π_m 就是最终的等价类集合。上述算法改为以任意顺序不断找出一个能被分裂的集合并不断迭代，同样也是正确的。

算法的时间复杂度为 $\mathcal{O}(n^2 \cdot |\Sigma|)$。

等价类划分算法

Algorithm 1: 等价类划分算法

Input: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Output: 等价类集合 $\Pi_0, \Pi_1, \Pi_2, \cdots$

$\Pi_0 \leftarrow \{Q \backslash F, F\}$;
$m \leftarrow 0$;
repeat
    $\Pi_{m+1} \leftarrow \Pi_m$;
    foreach $c \in \Sigma$ **do**
        $\Pi' \leftarrow \Pi_{m+1}$;
        foreach $G \in \Pi_{m+1} \wedge |G| > 1$ **do**
            if $\exists\, u, v \in G, \delta(u, c)$ 与 $\delta(v, c)$ 属于 Π_m 的不同组 **then**
                根据转移后不同的组，对 G 进一步划分得到 G^*;
                $\Pi' \leftarrow \Pi' \backslash \{G\} \cup G^*$;
            end
        end
        $\Pi_{m+1} \leftarrow \Pi'$;
    end
    $m \leftarrow m + 1$;
until $\Pi_m = \Pi_{m-1}$;

Myhill–Nerode 定理

对于语言 L 和一对串 x, y，定义等价关系 $x \equiv_L y$：对于任意串 z，$xz \in L$ 当且仅当 $yz \in L$。

Theorem

Myhill–Nerode theorem 语言 L 是正则的，当且仅当关系 $\equiv_L$ 的等价类个数有限。描述正则语言 L 的最小 DFA 唯一（忽略状态标号），且它的状态数量等于关系 $\equiv_L$ 对应的等价类数量。

从该定理的证明中，我们能看到：最小化后的 DFA 的状态一一对应等价类，最小化后的 DFA 就是唯一的最小 DFA。

Hopcroft 算法

Hopcroft 算法是一个寻找 DFA 等价类更加高效的算法，基于**启发式分裂**来优化时间复杂度。

该算法维护了用于划分等价类的**证据**集合，每次取出一个证据等价类 A，可以均摊 $\mathcal{O}(|\Sigma| \cdot |A|)$ 找到所有需要划分的等价类 Y。将等价类 Y 划分为等价类 U, V 之后，此时产生了新证据 U, V，我们能证明只需要保留证据 Y, U, V 中的两者仍然能保证正确性。于是，我们可以选择较小的证据并加入证据集合。每个状态每次作为证据出现，其所在的等价类大小至少除以二，所以算法的总时间复杂度为 $\mathcal{O}(|\Sigma| \cdot n \log n)$。

Hopcroft 算法

Algorithm 2: Hopcroft's algorithm

Input: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Output: 等价类集合 P

$P \leftarrow \{F, Q \backslash F\}$;

$W \leftarrow \{F\}$;

while $W \neq \emptyset$ **do**

  从 W 中任取一个集合 A，并将其从 W 中删去;

  foreach $c \in \Sigma$ **do**

    $X \leftarrow \{u \in Q \mid \delta(u, c) \in A\}$;

    foreach $Y \in \{u \in P \mid u \cap X \neq \emptyset, u \backslash X \neq \emptyset\}$ **do**

      $P \leftarrow P \backslash \{Y\} \cup \{Y \cap X\} \cup \{Y \backslash X\}$;

      if $Y \in W$ **then**

        $W \leftarrow W \backslash \{Y\} \cup \{Y \cap X\} \cup \{Y \backslash X\}$

      else

        if $|Y \cap X| \leq |Y \backslash X|$ **then**

          $W \leftarrow W \cup \{Y \cap X\}$;

        else

          $W \leftarrow W \cup \{Y \backslash X\}$;

        end

      end

    end

  end

end

FSM 在 OI 中的应用

虽然 FSM 是 OI 中相对冷门的内容，但是学习 FSM 有助于我们深入了解一些问题的本质，以及提出更加优秀的算法。接下来将略举几例来展现其丰富的应用。

一个例子

将 DP 套 DP 或者其它一些问题引入 DFA 的好处在于，我们可以使用有限状态自动机相关的理论来帮助我们解决问题。下面举出一个例子：

例 (Equanimous[1])

定义 $f(n)$ 表示将十进制数 n 所有数码之间填入加号或者减号，最终得到的值的绝对值最小值。

T 组询问，给定 l,r，对于所有 $k=0,1,\cdots,9$，求所有 m 之和满足 $l\le m\le r$ 且 $f(m)=k$。答案对 10^9+7 取模。

数据范围：$1\le T\le 10^4, 1\le l\le r\le 10^{100}$。

[1] XIX Open Cup Grand Prix of China, Problem E

DP 套 DP 与 DFA

考虑构造一个 DFA 能对于给定的 m 计算 $f(m)$。

令 $dp(i,j)$ 表示 m 的前 i 位插入加减号后能否变成绝对值为 j 的数。转移就是 $dp(i-1,j) \to dp(i,j+d_i), dp(i,|j-d_i|)$，其中 d_i 就是 m 的第 i 位数。

不难发现，DP 过程中状态可能会达到 $9 \cdot \log m$ 的级别，时间复杂度无法承受。实际上，我们能设定一个阈值 K，使得我们不需要考虑 $j \geq K$ 的 DP 信息，同样能够得到正确的结果。

我们记录 $0,1,\cdots,K-1$ 的 DP 信息，再在其之上 DP，这就是 **DP 套 DP** 的方法。

最小化 DFA

论文中证明了 K 取 $w(w-1)+1$ 即可满足条件（此处 $w=9$）。

我们将状态看做一个长度为 73 的 bitset，并搜索出所有可达的状态，这样的状态数量只有数万个。我们将状态对应的 f 值分为 10 类，并运行 Hopcroft 算法，最终的最小化 DFA 只有 715 个状态。

通过 DP 预处理足够的信息，我们能 $\mathcal{O}(10\cdot(\log l+\log r))$ 回答一次询问。

利用等价关系构造 DFA

例 (Median Replace Hard[2])

给定一个长度为 8 的二进制串 $P = P_0P_1\cdots P_7$。定义一个长度为 n 的二进制串 X 是好的，当且仅当能够通过执行 $(n-1)/2$ 次下述操作变为串 “1”：

- 选择 X 的连续三个比特 (X_i, X_{i+1}, X_{i+2})，将它们替换为 P 的第 $(X_i + 2X_{i+1} + 4X_{i+2})$ 个比特。

共 T 组询问。给定一个包含 0,1,? 的串 S，问存在多少个将 ? 替换为 0,1 的方案，使得最后的串为好串。答案对 10^9+7 取模。

数据范围：$1 \le T \le 256, 1 \le |S|, \sum |S| \le 300000$，$|S|$ 为奇数

[2] XX Open Cup Grand Prix of Tokyo, Problem J

利用等价关系构造 DFA

考虑构造一个 DFA 来识别所有的好串。

回忆 Myhill–Nerode 定理中的等价关系，根据等价类直接构造出 DFA。但枚举所有的串 z 来判断等价性是不可能的，考虑仅枚举长度不超过 L 的串 z。可以证明本题取 $L = 10$ 满足条件。

利用等价关系构造 DFA

考虑构造一个 DFA 来识别所有的好串。

回忆 Myhill–Nerode 定理中的等价关系，根据等价类直接构造出 DFA。但枚举所有的串 z 来判断等价性是不可能的，考虑仅枚举长度不超过 L 的串 z。可以证明本题取 $L = 10$ 满足条件。

不妨将所有等价类中任意长度最小的串视作代表元，我们可以使用 BFS 找出所有的代表元。先将空串入队，每次从队首弹出一个串 x，并判断串 x 和已有的代表元是否等价。如果均不等价，它能成为某个等价类的代表元。不难证明这个算法能找到所有等价类，观察这个过程也能发现所有代表元形成了一个树形结构。

OI 字符串理论中的 DFA

DFA 自身的特性使它成为 OI 中的许多字符串算法的基础结构。这其中成果最为丰富，应用最为广泛的当属后缀数据结构。

后缀 Trie 是最基本的结构，在此基础上进行压缩（缩去出入度均为 1 的点）就得到了后缀树，而后缀自动机就是最小化后的后缀 Trie。同时对后缀 Tire 进行最小化和压缩能够得到压缩后缀自动机等更进一步的结果。从 DFA 的角度思考，有助于我们更加深刻地理解这些结构。

另一个例子

接下来的这道例题需要掌握后缀自动机的基础知识。

例 (Beautiful Automata[3])

构造一个仅包含小写字母的串 s，使得 s 的后缀自动机的转移图（仅保留其有向图的结构）与给定的 DAG 同构。如果不存在，输出 -1，否则输出字典序最小的解。

数据范围：$1 \leq n \leq 2000, 1 \leq m \leq 3000$。

[3] XIX Open Cup Grand Prix of Baltic Sea, Problem G

解答

入度出度为零的唯一状态设为 S, T。SAM 中状态 u 是 right 集合相同的串的集合，所有从 S 到 u 的路径长度构成了一个区间。设从 S 到 T 的路径长度为 $K+1, K+2, \cdots n$，代表从 $1, 2, \cdots, n-K$ 开始的后缀。根据这 $n-K$ 条路径的第一条边，能贪心确定 s 的前 $n-K$ 个字符。

枚举 T 的后缀连接 P，若存在 P 到 T 的路径长度为 l，那么 $S[n-K+1:n] = S[n-K+1-l:n-l]$，可以唯一确定整个串 s。考虑恢复 DAG 的转移边字符。从 T 出发的任意一条路径构成了 s 反串的前缀。从 T 开始 BFS 能唯一确定 DAG 的转移边。与 s 的真实 SAM 比较即可。时间复杂度为 $\mathcal{O}(nm|\Sigma|)$。

正则表达式匹配的算法

正则表达式匹配的常见做法就是将其转化为 NFA 的计算问题。除了套用一般化的算法，利用“Thompson 构造法每次只会增加常数条转移边”的性质，我们可以得到一个时间复杂度为 $\mathcal{O}(ns)$ 的算法。

实际上，通过分析 Thompson 构造法得到的 NFA 的性质，并运用 Method of Four Russians（对正则表达式树按 $\Theta(\log n)$ 大小分块，建立嵌套的 NFA 结构），我们能做到 $\mathcal{O}(\frac{ns}{\log n})$ 的时间复杂度。

总结

本次交流简述了有限自动机的基础理论及算法，并展现了相关理论在信息学竞赛中的丰富应用。

有限状态自动机是一个非常简单的结构，但是我们能发掘大量的性质与许多有趣的应用，这就是它的魅力所在。我相信，有限状态自动机仍然有着巨大的潜力，等待我们去进一步探索更加有趣的理论与应用。希望本次交流能起到抛砖引玉的作用，吸引更多的选手来学习和研究有限状态自动机。

致谢

谢谢大家，欢迎提问！