第3章 有穷自动机

DFA

形式化描述：

- A formalism for defining languages, consisting of:
  1. A finite set of *states* (Q, typically).
  2. An *input alphabet* (Σ, typically).
  3. A *transition function* (δ, typically).
  4. A *start state* (q_0, in Q, typically).
  5. A set of *final states* ($F \subseteq Q$, typically).
     - “Final” and “accepting” are synonyms.

迁移函数是一个全函数，如果没有下一个状态的话则加入死状态。

特点：对于某一特定输入，其下一个状态是确定的（有且仅有一个）

可以用两种方法表示DFA：

1. 图：初始状态用箭头指向，接收状态用双圈表示
2. 迁移表：初始状态用箭头指向，接收状态用 * 标记

DFA的语言：L(A) = the set of strings w such that δ(q0, w) is in F

判断DFA所接收的语言：就是判断集合等价，需要双向证明属于关系

NFA

形式化定义：

- A finite set of states, typically Q.
- An input alphabet, typically Σ.
- A transition function, typically δ.
- A start state in Q, typically q_0.
- A set of final states $F \subseteq Q$.

特点：对于某一特定输入，其下一个状态是一个集合（有多种可能）

假设：the NFA always “guesses right.”

ε-NFA

多了在ε上的操作，求闭包：CL(q) = set of states you can reach from state q following only arcs labeled ε

扩展迁移函数：

Intuition: $\hat{\delta}(q, w)$ is the set of states you can reach from q following a path labeled w.

Basis: $\hat{\delta}(q, \epsilon) = CL(q)$.

Induction: $\hat{\delta}(q, xa)$ is computed by:

1. Start with $\hat{\delta}(q, x) = S$.
2. Take the union of $CL(\delta(p, a))$ for all p in S.

正则语言

定义：A language L is *regular* if it is the language accepted by some DFA.

常见的非正则语言：

L_1 = {0^n1^n | n ≥ 0}

L_2 = {w | w in {(,)}* and w is *balanced* }

DFA和NFA的等价性

DFA->NFA：转换成下一状态的集合只包含一个状态的NFA

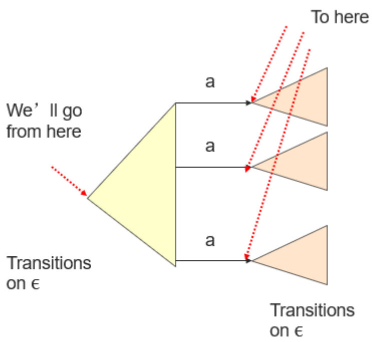
NFA->DFA：子集构造法

NFA和ε-NFA的等价性

NFA->ε-NFA：NFA本身就是ε-NFA

ε-NFA->NFA：

对迁移函数做如下转换：



- Compute $\delta_N(q, a)$ as follows:
  1. Let S = CL(q).
  2. $\delta_N(q, a)$ is the union over all p in S of $\delta_E(p, a)$.
- F' = the set of states q such that CL(q) contains a state of F.