Does the black moon howl?

一，String

字母表 Alphabet 是任何非空有限集。

长度为零的字符串称为空字符串，写为ε

翻转reverse w = 123 w^R = 321

w^k+1 = ww^k

w^0 = ε

前缀prefix

We say that string x is a prefix of string y is there exists a string z such that y = xz and denoted by x⊆y.

We say x is a proper prefix of y if there exists a string z such that z≠ε and y = xz and denoted by x⊂y

A language is prefix-free if no member is a proper prefix of another member.

Kleene Star

If A is a language then the Kleene Star of A is denoted by A^∗ and is define by:

A^∗ = U_k≥0 A^k

For singletons instead of {a}^∗ we write a^∗ .

By A^+ we mean A^∗ / {ε}

比如 A= {a,b}

A^0 = {ε}

A^1 = A

A^2 = {aa,ab,ba,bb}

A^3 = {aaa,aab,aba,baa,abb,bab,bba,bbb}

A* = {ε,a,b,aa,ab,ba,bb,aaa,.....}

A^+ = {a,b,aa,ab,ba,bb,aaa,.....}

二，有限自动机 finite automaton （FA）

if Σ={p, q,→, ¬} then A={p→q,→p¬q, ¬q→¬p} is a language over Σ

有穷状态机 finite automaton （FA)

有穷自动机（finite automaton）是最简单的计算模型（computational model），又叫有穷状态机（finite state machine）；对应马尔科夫链（Markov chain）。

能被FA识别的语言就是正则语言 Regular Languages

识别语言的格式 Recognizing languages

L(M) 表示一种语言M

Say that A is the language of M1 and that M1 recognizes A and that A = L(M1)

L(M) = {w| M accepts w}

L(M) is the language of M

M recognizes L(M)

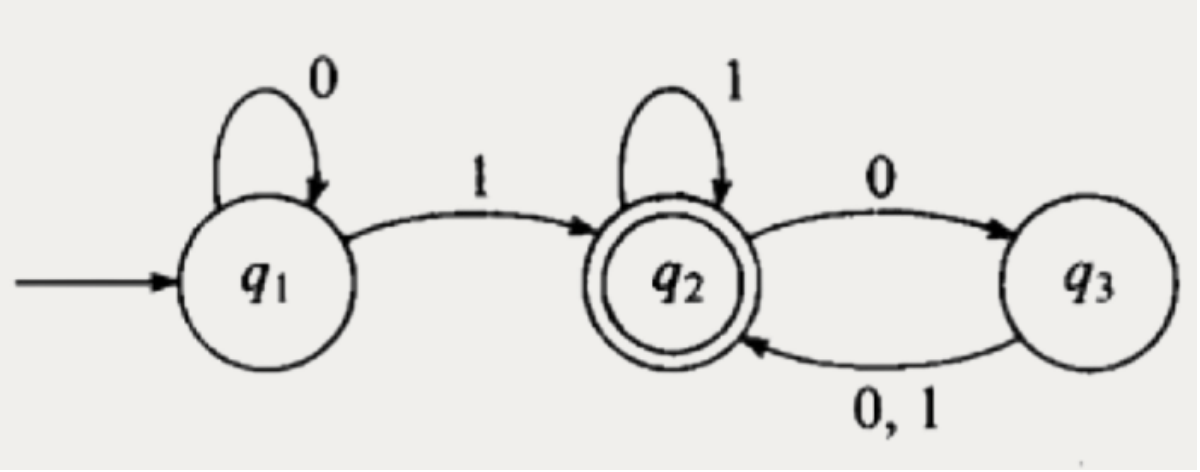
句子和语言的关系 Strings and languages

The empty string ε is the string of length 0

The empty language ø is the set with no strings

如何描述 finite automaton

- 状态集（state set）Q，包含有限种可能的状态
- 字母表（alphabet）Σ，包含所有可能的输入符号
- 转移函数（transition function）δ，描述从任意属于Q的状态，经Σ中的输入，映射到Q 中任意状态的函数，可以用转移表（transition table）来描述，记$Q \times \Sigma \rightarrow Q$
- 起始状态（start state）$q0 \in Q$
- 接受状态集（accept state set），又称终止状态集（final state set） $F \subseteq Q$，包含一系列可行的终止状态



即，我们可以使用5元组(Q,Σ,δ,q0,F)来描述一个有穷自动机。以上面的状态图为例，其5元组为：

- Q={q1,q2,q3}
- Σ={0,1}
- δ=

  | 0 1

——————

q1 | q1 q2

q2 | q3 q2

q3 | q2 q2

- q1
- F={q2}

正则语言和正则表达式

能被FA识别的语言就是正则语言

正则表达式显示等价于有限自动机

正则语言的闭包性

4个正则表达式符号 ∪, ∘, ∗,∩ 三个集 Σ, ∅,ε

并集、交集、连接、克林闭包

Union: $A \cup B=\{w \mid w \in A \text{ or } w \in B\}$

Concatenation: $A \circ B=\{xy \mid x \in A \text{ and } y \in B\}=AB$

Star: $A^*=\{x_1 \ldots x_k \mid \text{each } x_i \in A \text{ for } k \geq 0\}$

Note: $\varepsilon \in A^*$ always

$(0 \cup 1)^*=\Sigma^*$ gives all strings over Σ

Σ^*1 gives all strings that end with 1

$\Sigma^*11\Sigma^*$= all strings that contain 11 =L(M1)

Example. Let A={good, bad} and B={boy, girl}

A∪B= {good, bad, boy, girl}

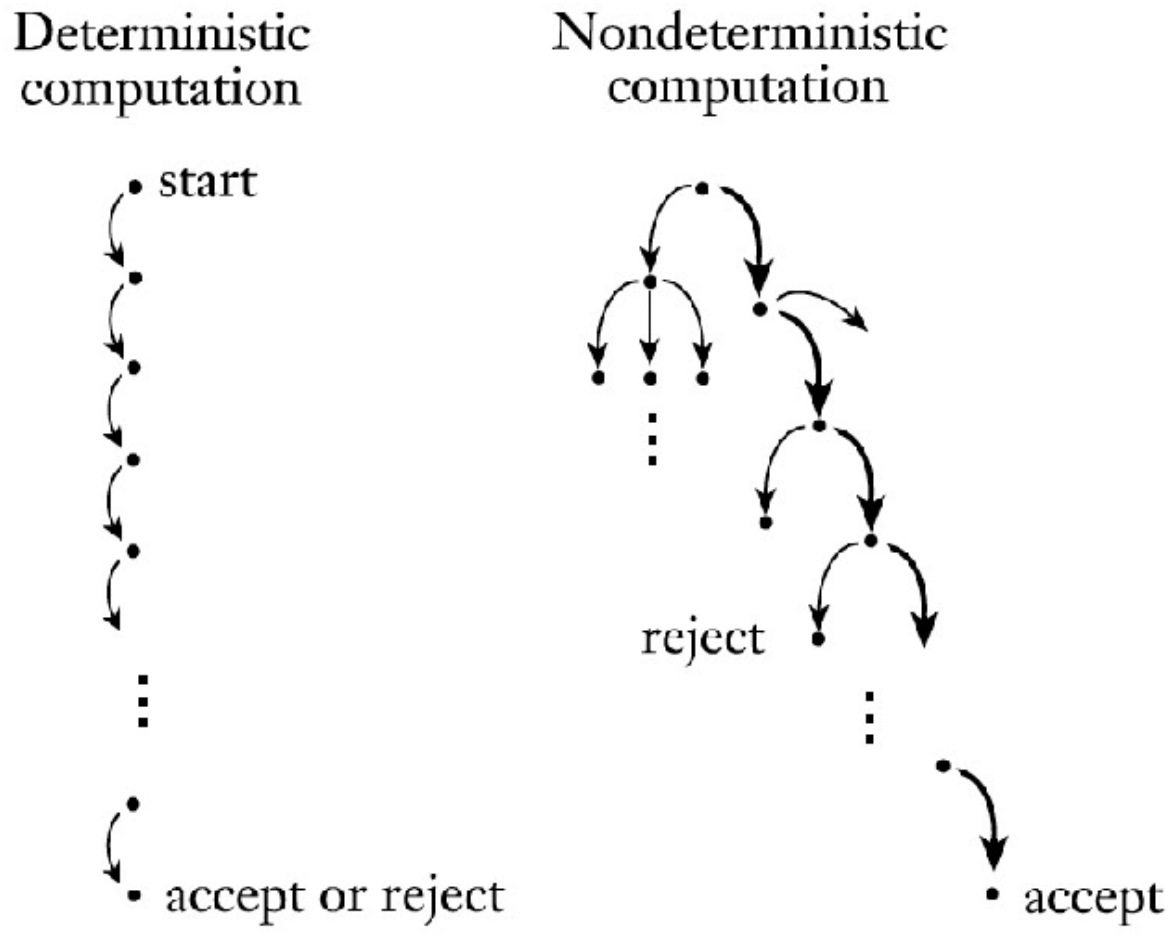
A∘B=AB= {goodboy, goodgirl, badboy, badgirl}

A^∗= {ε, good, bad, goodgood, goodbad, badgood,badbad, goodgoodgood, goodgoodbad, ... }

正则语言的闭包性是指正则语言在特定操作下能够保持其正则性的一种性质。具体来说，正则语言对于并集、连接、闭包和补集等操作是封闭的，即通过这些操作得到的新语言仍然是正则语言。

三，不确定有穷自动机 Nondeterministic Finite Automata （NFA）

NFA和DFA的区别：NFA允许匹配 ε

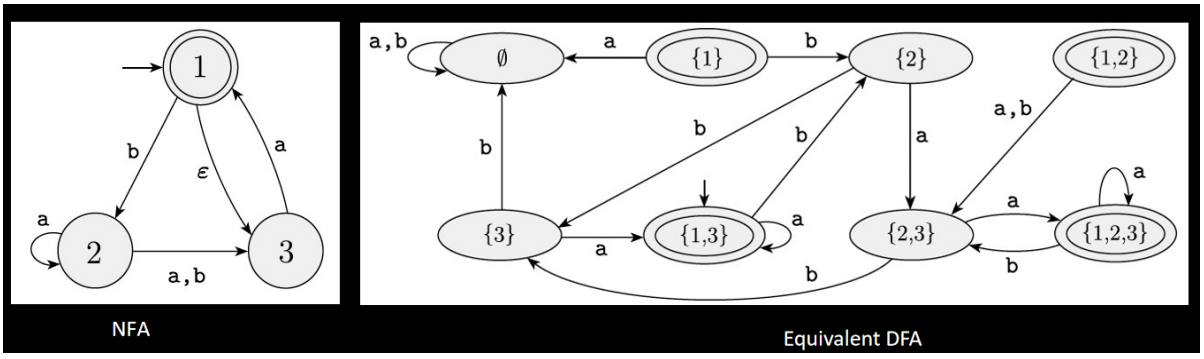


NFAs和DFAs的等价性

每个非确定性有限自动机 NFA 都有一个等价的确定性有限自动机 DFA

对于有n个状态的NFA，等价的DFA则有2^n个状态

construct eight states, one for each subset of NFA's states, label each of D's states with the corresponding subset as{∅, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}



Regular Expressions的一些判别

R is a regular expr if：

R is a for some a in the alphabet Σ,

ε

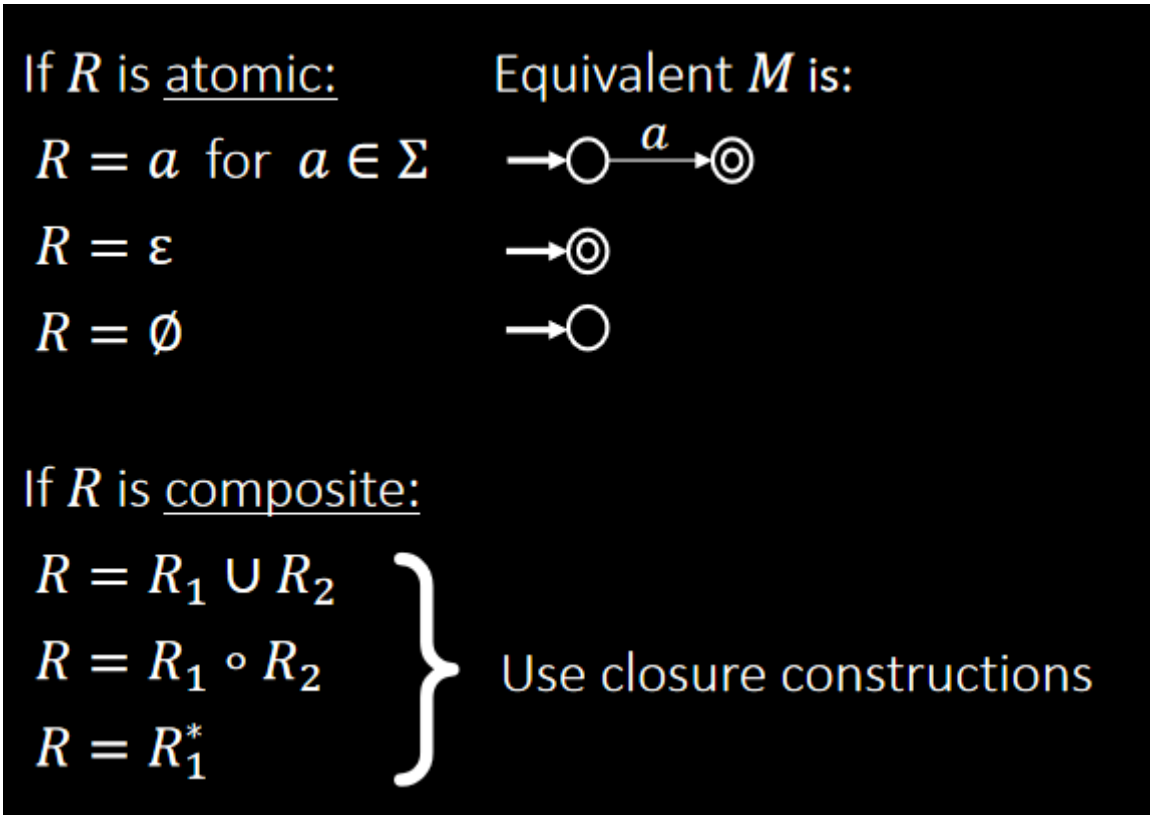
Ø

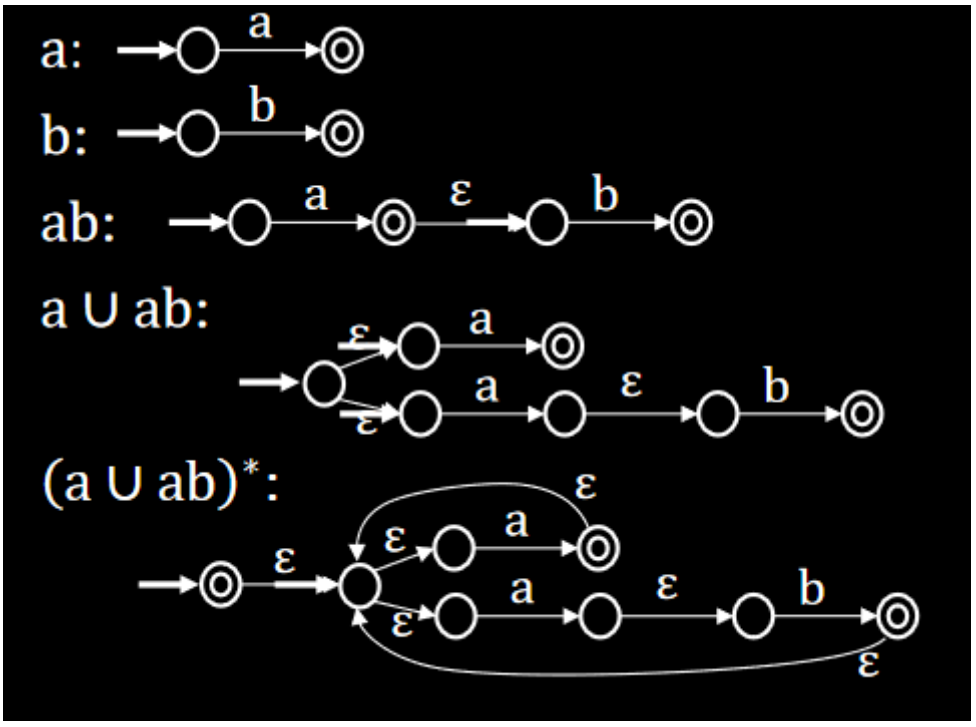
R=R_1∪R_2, where R_1 and R_2 are regular expressions,

R=R_1∘R_2, where R_1 and R_2 are regular expressions,

R_1^∗, where R_1 is regular expression,

如何根据正则表达式构造NFA

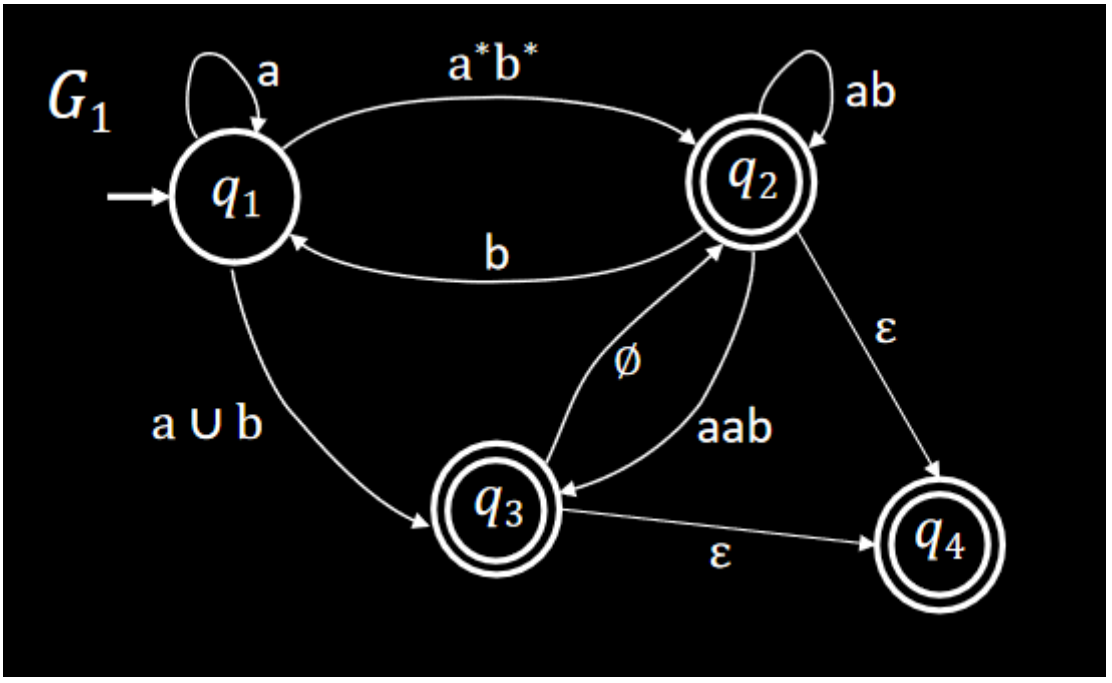




广义非确定性有限自动机（GNFA）

A Generalized Nondeterministic Finite Automaton (GNFA) issimilar to an NFA, but allows regular expressions as transition labels

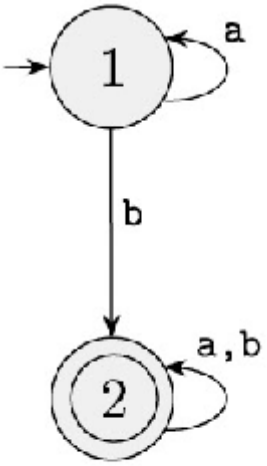
广义非确定性有限自动机（GNFA）类似于NFA，但允许正则表达式作为转换标签



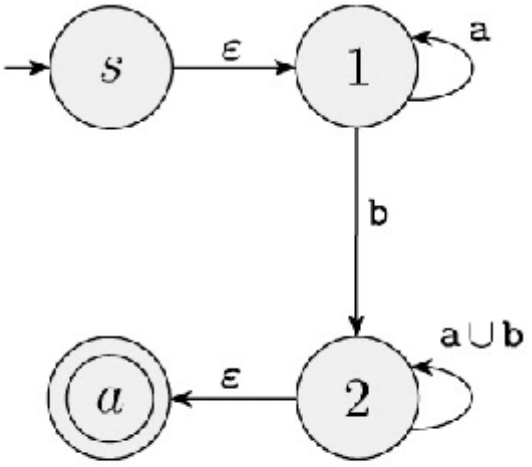
如何把DFA → Regular Expressions

把DFA->GNFA-> (k-1)GNFA-> Regular Exp

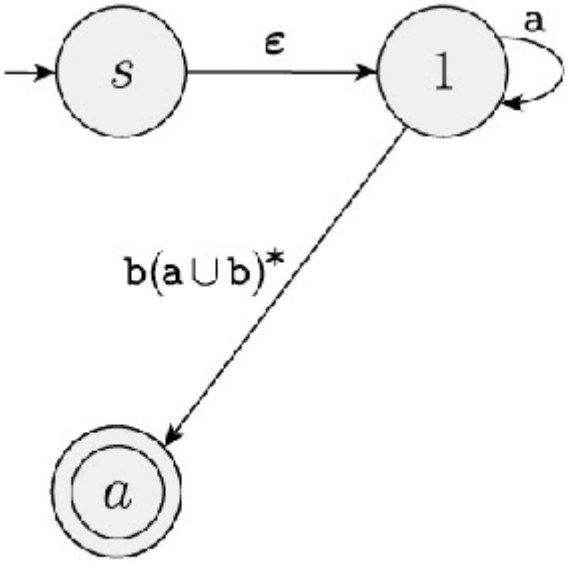
例子1


1
a
b
2
a,b

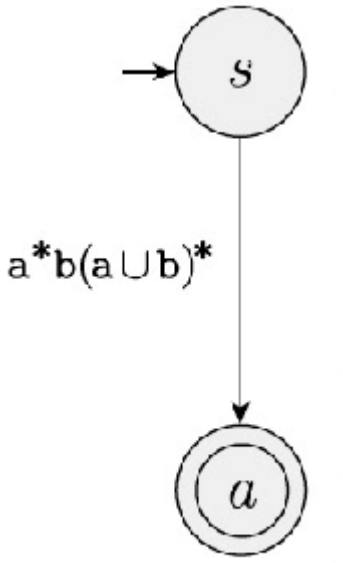
(a)


s
ε
1
a
b
a
ε
2
a∪b

(b)

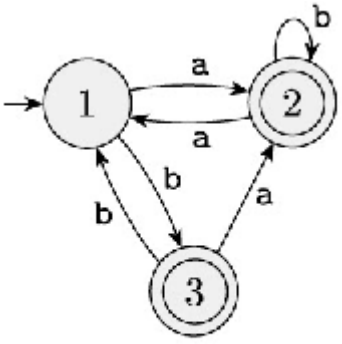

s
ε
1
a
b(a∪b)*
a

(c)

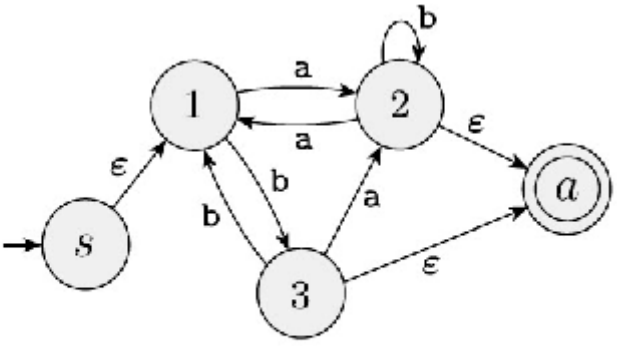

s
a*b(a∪b)*
a

(d)

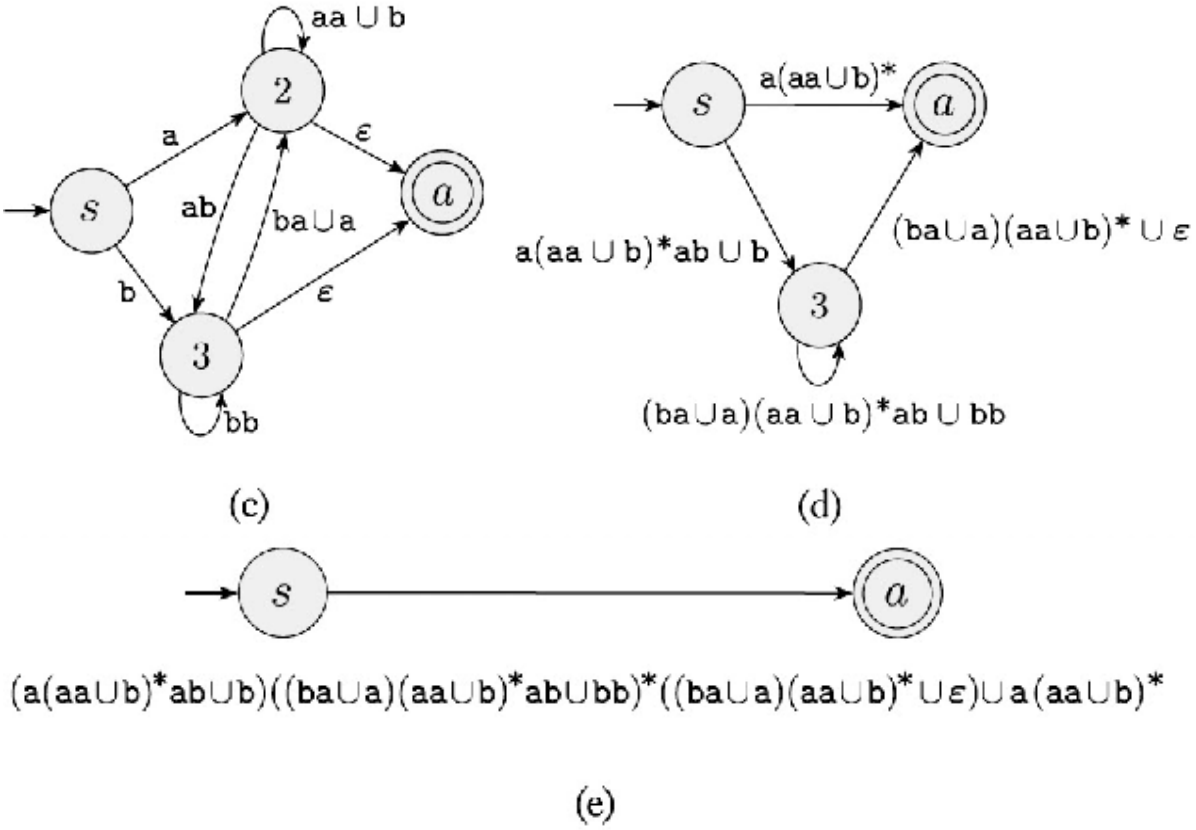
例子2


1
a
2
b
a
b
b
a
3

(a)


1
a
2
b
a
ε
ε
b
b
a
s
3
ε
a

(b)



如何证明一个语言不正则 Non-Regular Languages 泵引理 Pumping Lemma

泵引理指出所有的正则语言都有一种特殊的性质，如果能够反证某个语言不具有该性质，则可以证明该语言不是正则的。

泵引理 Pumping Lemma：

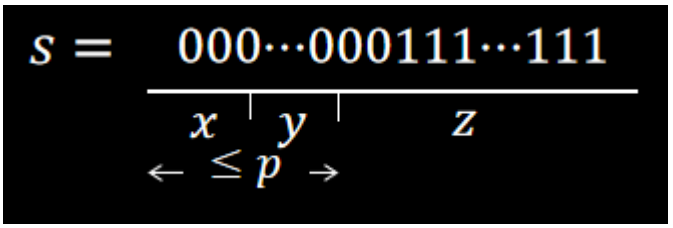
若A为正则语言，则存在数p，称泵长度（pumping length），使得A中任意长度不小于p的字符串s，都可以被分为3段，s=xyz，且满足：

- ∀i≥0 , xy^i z ∈ A　　y^i = yyyy....
- y != ε
- |xy| ≤ p （|xy| 表示字符串xy的长度）

泵引理例子

Let D = {0^k 1^k | k≥0}

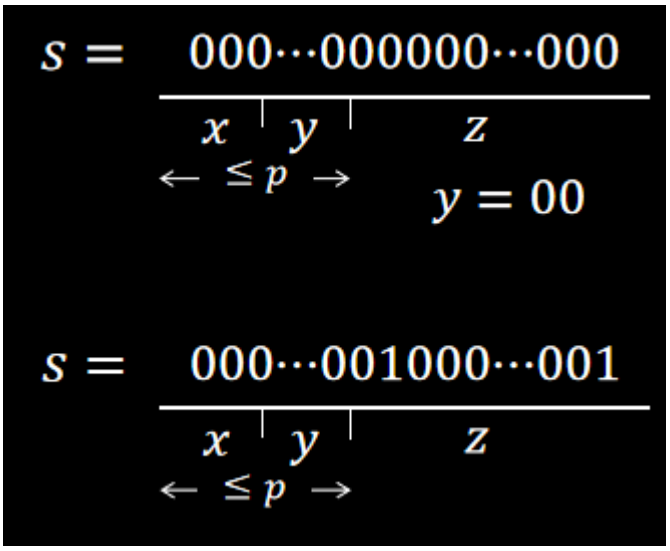
1. Assume that D is regular.
2. Let s = 0^p 1^p　∈D
3. Pumping lemma says that can divide s=xyz satisfying the 3 conditions.



4. But xyyz has excess 0s and thus xyyz∉D
5. So, D is not regular.

Let F={ww| w∈Σ^∗} . Say Σ^∗={0,1}.

1. Assume (for contradiction) that F is regular.
2. Let s=0^p 0^p ∈F
3. Pumping lemma says that can divide s=xyz satisfying the 3 conditions.



4. Try s=0^p 0^p ∈F. But that s can be pumped and stay inside F. Bad choice.
5. Try s=0^p 10^p 1 ∈F. Show cannot be pumped s=xyz satisfying the 3 conditions.xyyz∉F Contradiction!
6. Therefore F is not regular.

将闭合特性与泵引理结合的证明一个语言不正则

Let B={w| w has equal numbers of 0s and 1s}

Show: B is not regular

Proof by Contradiction:

Assume (for contradiction) that B is regular.

We know that 0^∗ 1^∗ is regular（因为这就特么的是一个正则表达式） so B ∩ 0^∗ 1^∗ is regular.（闭包性）

But D= B ∩ 0^∗ 1^∗ and we already showed D is not regular.

Contradiction! Therefore our assumption is false, so B is not regular.

四，上下文无关语言 Context Free Languages

上下文无关语法 Context Free Grammars (CFGs)

如果一个语言能够被**上下文无关文法**（context-free grammar, CFG）接受，则称该语言为**上下文无关语言**（context-free language, CFL），对应**下推自动机**（pushdown automaton, PDA)

上下文无关文法的形式化定义为一个4元组 **(*V*,Σ,*R*,*S*)**，分别为：

- **V**，有穷的**变元**（variable）集合；
- **Σ**，与V不相交的符号集，称**终结符集**（terminals）；
- **R**，有穷**规则集**（rules）；
- **S**，**起始变元** $S \in V$

例子：

G2：

E → E+T | T

T → T×F | F

F → (E) | a

V = {E, T, F}

Σ = {+, ×, (,), a}

R = the 6 rules above

S = E

G0

A -> 0A1

A -> B

B -> #

Grammar G_0 contains three rules.

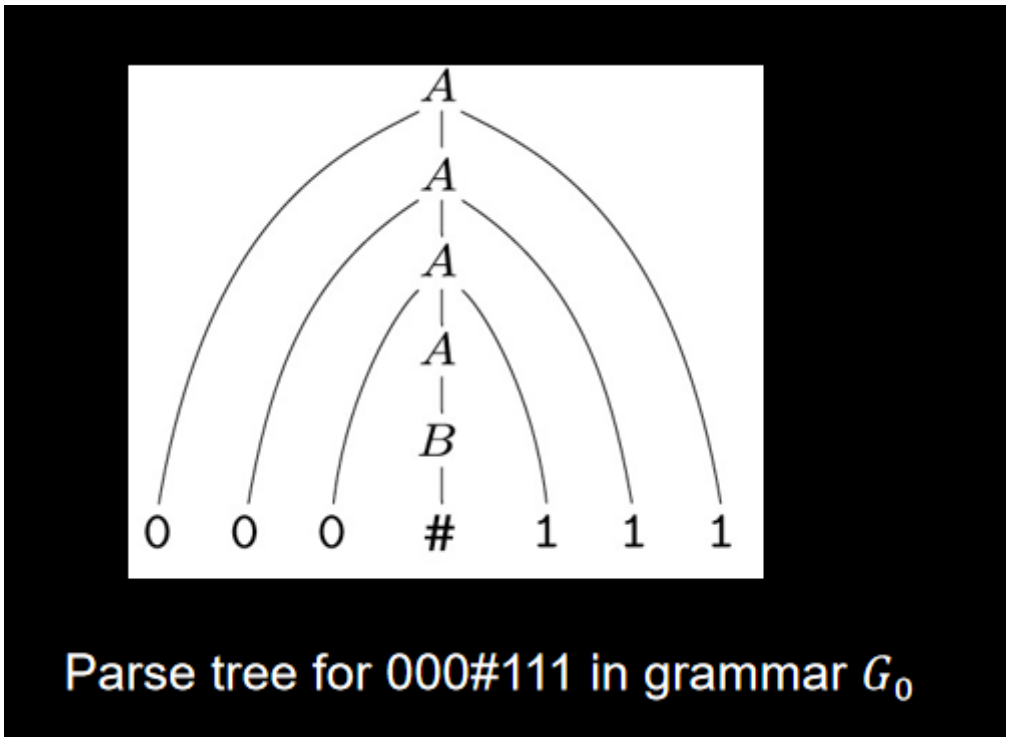
Its variables are A and B, where A is the start variable.

Its terminals are 0, 1, and #.

CFG的语法分析树

对上节语法分析

A ⇒ 0A1 ⇒ 00A11 ⇒ 000A111 ⇒ 000B111 ⇒ 000#111.



CFG例子2

L(G1) = {0^k1^k| k≥0}

G1

S -> 0S1

S -> R

R -> ε

Shorthand:

S -> 0S1 | R

R -> ε

CFG的模糊 ambiguous

If a string has two different parse trees then it is derived ambiguouslyand we say that the grammar is **ambiguous**.

如果一个字符串有两个不同的解析树，那么这个CFG语法是模糊的。

下推自动机 Pushdown Automata (PDA)

相较于有穷自动机，下推自动机拥有一个能够存储的栈，因而拥有更强的表现能力。

例如，DFA无法识别 {0^n 1^n|n≥0}，但下推自动机能够通过栈记忆0的个数，从而接受该语言。

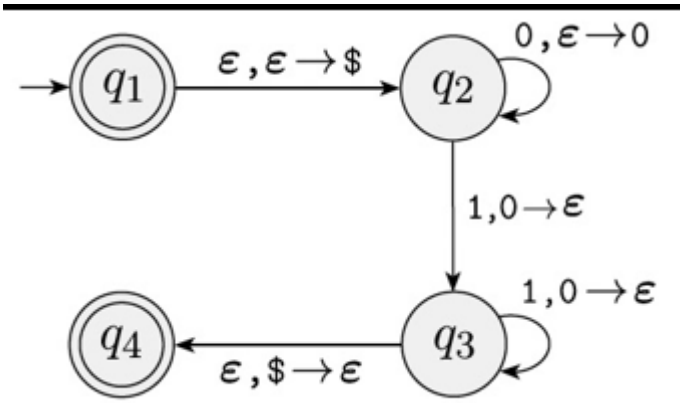
下推自动机（pushdown automata, PDA）可以被表示为6元组(Q,Σ,Γ,δ,q0,F)：

- Q 为状态集
- Σ 为输入字母表

- Γ 为栈字母表（stack alphabet）
- δ 为转移函数
- q0 ∈ Q 为初始状态
- F ⊆Q 为接受状态集

例子： PDA for D={0^k1^k | k≥0}

图中的ε可以理解为表示栈顶的元素，$表示栈底标识，ε -> 0 就代表把栈顶的ε变成0， 0 -> ε 表示把栈顶的0变成ε



Q = {q_1,q_2,q_3,q_4},

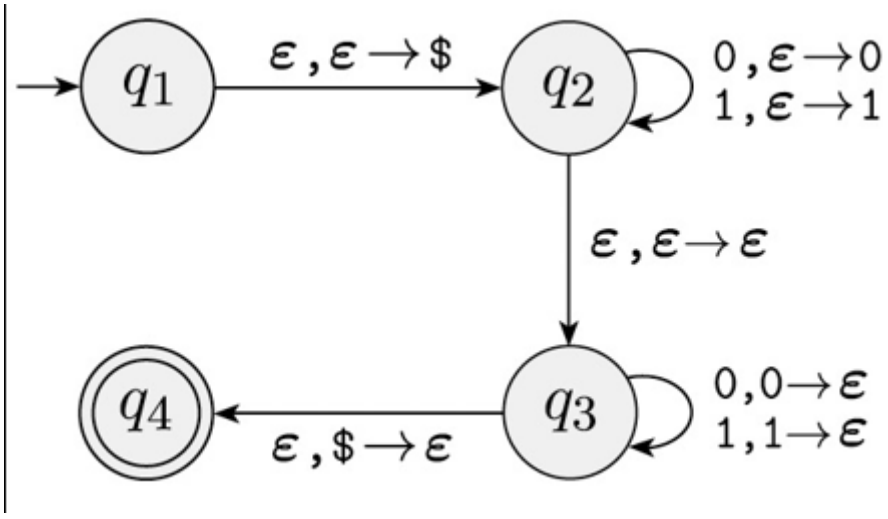
Σ = {0,1},

Γ = {0, $},

F = {q_1,q_4},

and δ is given by the following table, wherein blank entries signify ∅.

例子： B = {ww^$\mathcal{R}$ | w∈{0,1}^∗ }



将CFG转换为PDA

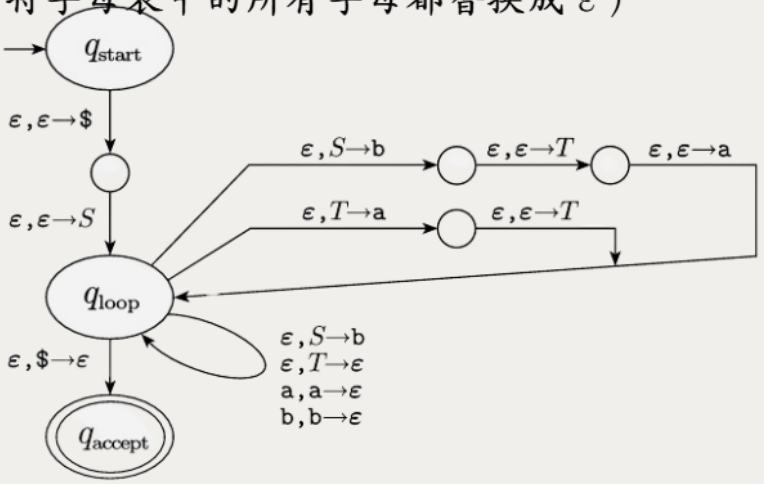
【计算理论】上下文无关语法 (CFG) 转为 下推自动机 (PDA) cfg构造pda-CSDN博客

例题：将下述CFG转化为PDA

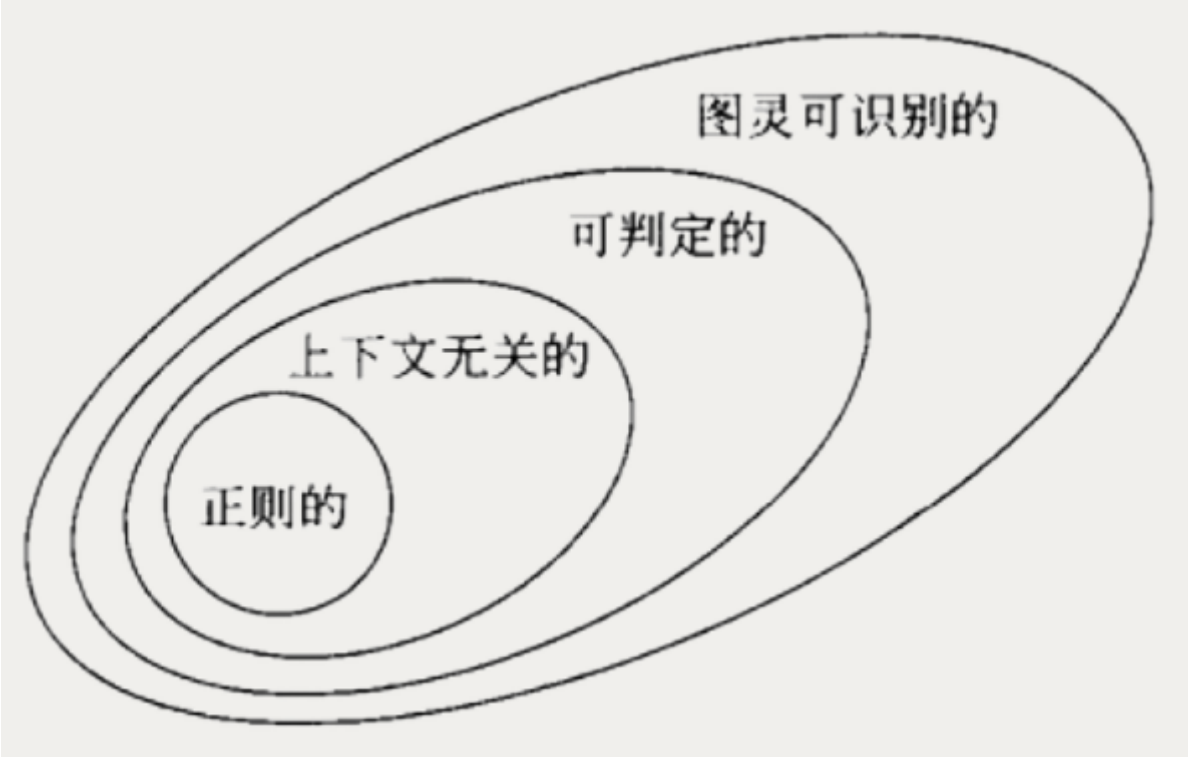
$$S \rightarrow aTb \mid b$$
$$T \rightarrow Ta \mid \varepsilon$$

思路：

1. 画出qstart,qloop,qaccept
2. 写压栈条件，即$和S
3. 写loop条件，先写倒序的所有展开式
4. 写accept条件（即 将字母表中的所有字母都替换成ε）



CFG和PDA的等效性



用泵引理证明一个语言不是上下文无关语言

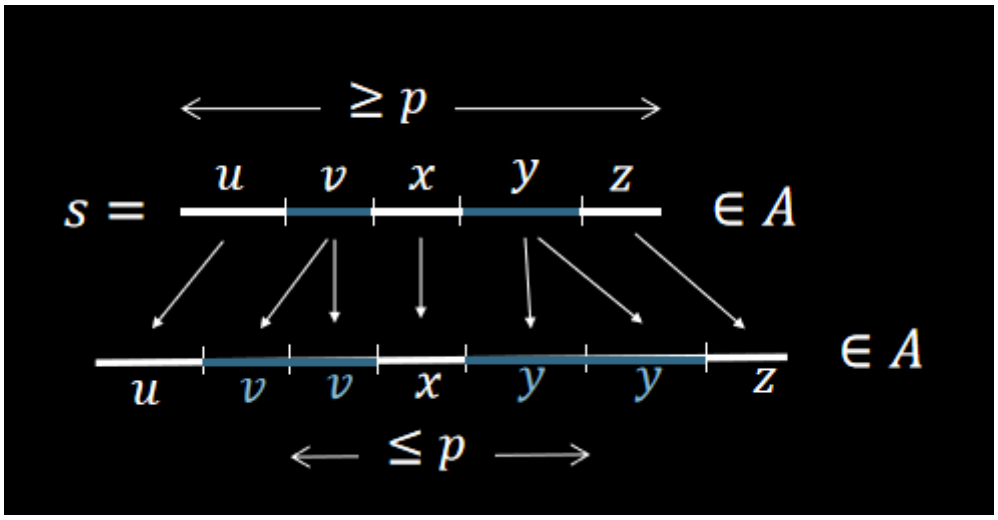
若A为CFL，则存在数p，为泵长度，使得A中任意长度不小于p的字符串s，都可以被划分为5段s=uvxyz，且满足：

∀i≥0 , u v^i x y^i z ∈A

|vy|>0

|vxy|≤p

例子： Let B={0^k 1^k 2^k | k≥0}.

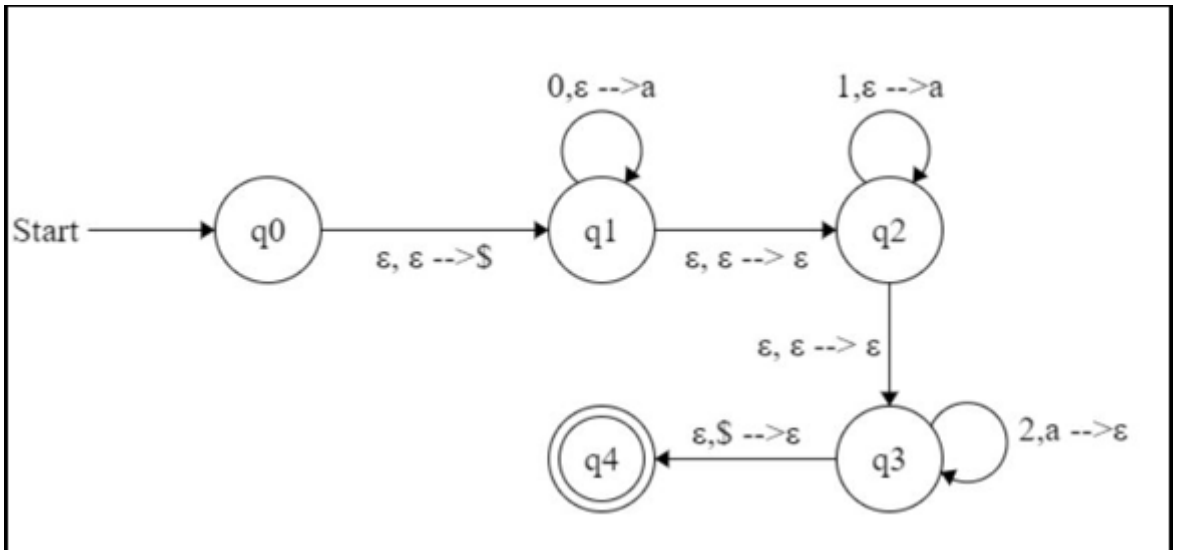


一个练习

Let L={0^i1^j2^k| i+j=k}, construct a CFL and PDA for it.

S→ 0S2| B

B → 1B2| ε



五，图灵机

图灵机（Turing machine）与DFA类似，但相比DFA，图灵机具有无限制的存储空间，且读写头可以左右移动、既能读也能写。并且，与DFA不同，当图灵机进入accept或者reject状态时，图灵机会立即停机。

图灵机可以模拟所有实际计算机的所有计算行为，但仍存在不可解的问题。

一台图灵机可以被形式化描述为一个7元组：{Q,Σ,Γ,δ,q0,qaccept,qreject}.

- 状态集Q
- 字母表Σ
- 带子Γ
- 转移函数δ，如　　是读到0用空白字符替换，读写头向右走一格
- 起始状态q0
- 接受状态q_accept
- 拒绝状态q_reject

例子： A = {0^2^n | n ≥ 0}

The formal definition: M2=(Q, Σ, Γ, δ,q_0, qacc , qrej)

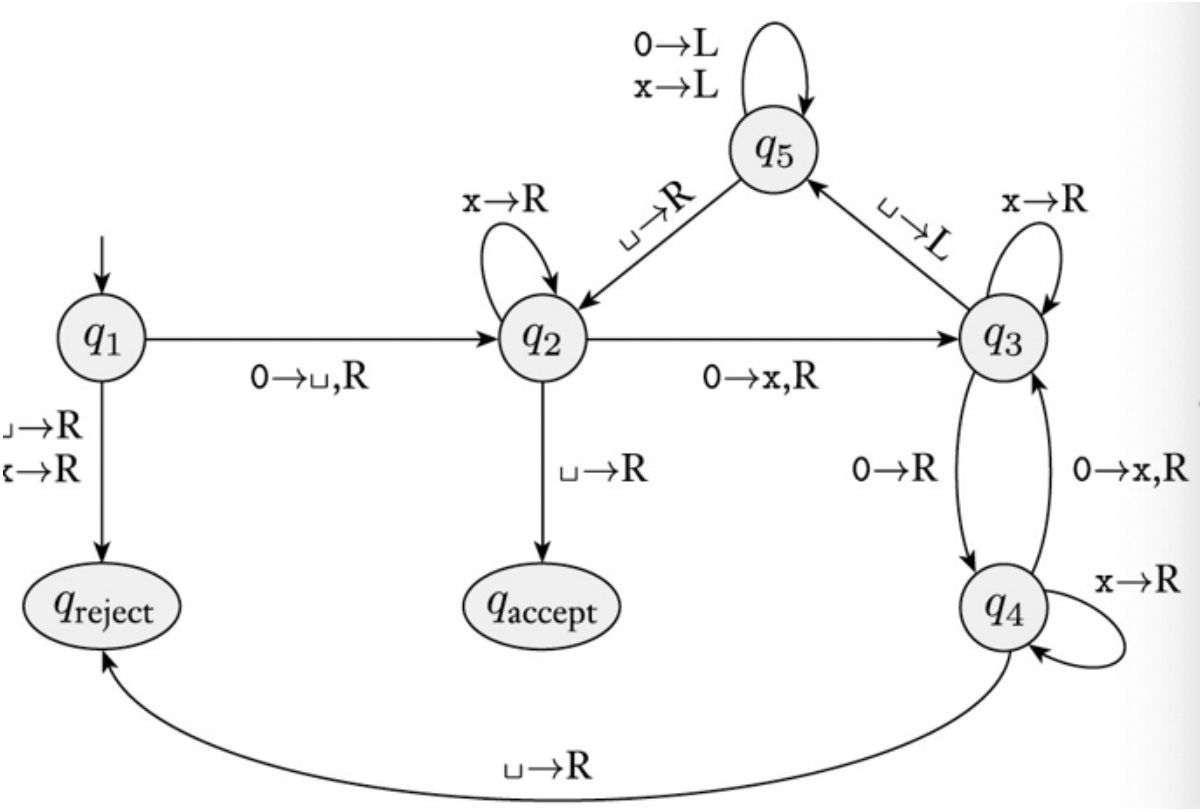
Q={q_1,q_2,q_3,q_4,q_5, qacc , qrej}

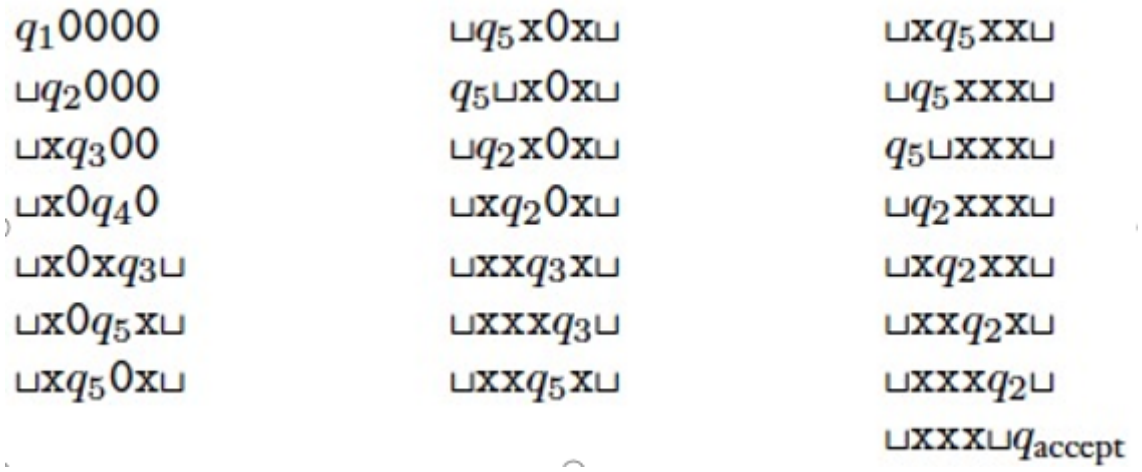
Σ={0}

Γ={0, x,␣}

We describe δ with a state diagram.

The start, accept, and reject states are q_1, q_acc , q_rej





语言描述法：

M2 = “On input string w:

1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1 the tape contained a single 0, accept.
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, reject .
4. Return the head to the left-hand end of the tape. 5. Go to stage 1.”

【计算理论】04 图灵机 用语言描述判定语言的图灵机-CSDN博客

多带图灵机

具有多条带子的图灵机，转移函数允许多个带子同时进行读、写和移动读写头。

易证明：任何多带图灵机，都可被转化为一台单带图灵机（等价性）。

确定/不确定图灵机

类似非确定状态机和下推自动机，如果图灵机在计算过程中，可以非确定的选择多种可能动作中的一个，就称该图灵机为非确定的。非确定形图灵机的转移函数符合δ:Q×Γ→P(Q×Γ×{L,R})。

每个非确定图灵机都等价于某个确定型图灵机。

枚举器

xxx

Church-Turing Theory

xxx

六，可判断性

recognizable称一个语言是可识别的，当且仅当存在一台识别该语言的图灵机。

decidable 称一个语言是可判定的，当且仅当存在一台识别该语言的图灵机，该图灵机一定会停机，称为判定器（decider）。

描述可判定性/可识别性的目的在于，这些性质反映了哪些问题能够通过算法求解。

A_DFA是可判定的

一定有且仅有一台图灵机可以判断字符串w是否能被DFA B接受

A_DFA={<B,w>|B为DFA，且B接受w}

***A_NFA*,*A_REX*是可判定的**

一定有且仅有一台图灵机可以判断字符串w是否能被NFA B接受（通过转化为DFA）

一定有且仅有一台图灵机可以判断字符串w是否能被正则语言 R接受（通过转化为DFA）

ANFA={<*B*,*w*>|*B*为*NFA*，且*B*接受*w*}

AREX={<*R*,*w*>|*R*为正则表达式，且*R*能够派生*w*}

***E_DFA*是可判定的**

一定有且仅有一台图灵机可以判断此DFA的语言是否是空集

EDFA={<*A*>|*A*是一个*DFA*，且*L*(*A*)=∅}

EQ_DFA是可判定的

一定有且仅有一台图灵机可以判断这两DFA是不是一样的

EQ_DFA={<A,B>|A,B为DFA，L(A)=L(B)}

***A_CFG*是可判定的**

ACFG={<G,w>|G是CFG，w是串，G派生w}

E_CFG可判定

一定有且仅有一台图灵机可以判断一个cfg是不是只能生成语言为空集的语言

E_CFG={<*G*>|*G*为*CFG*，且*L*(*G*)=∅}

***EQ_CFG*不可判定 (因为上下文无关语言类对交运算和补运算不封闭)**

EQ_CFG={<*G*,*H*>|*G*,*H*都是*CFG*，且*L*(*G*)=*L*(*H*)}

A_TM不可判定

ATM={<M,w>|M为一台图灵机，M接受w}

***A_TM*可识别(可包含不停机)**

由于*A_TM*可识别、不可判定，且可判定语言A 与其补A ¯都是可识别的，可知 ¯*A_TM*不可识别，得证存在不可被图灵机识别的语言。

所有串构成的集合Σ*可数

因为Σ可数，则Σ*中长度为n的字符串有|Σ|^n个（每个位置可以是Σ中的任意一个字符）

因此，Σ*中所有字符串的数量可以通过对所有长度的字符串数量求和来得到：Σ_{n=0}n。

这是一个几何级数，其和是有限的（因为|Σ|是一个有限数），并且与自然数集有双射关系

所有图灵机构成的集合可数

所有图灵机构成的集合之所以是可数的，原因在于每个图灵机都可以通过一个独特的编码来表示，这个编码是一个有限的字符串。由于字符串的有限性和可枚举性，我们可以为每一个图灵机分配一个唯一的自然数，从而与自然数集建立双射关系。

所有语言集合不可数

引证：所有无限二进制序列的集合B是不可数的。

（证明：课本有，作业也有）

因为图灵机构成的集合是可数的，而字母表上所有的语言构成的集合是不可数的，两者不能对应；可知一定存在不能被任何图灵机识别的语言。

补图灵可识别与可判定的关系

定理：对任意语言A，该语言可判定，当且仅当*A*与 都可识别。

八，可归约性

[可归约性在计算理论与密码学的应用-CSDN博客]

映射可归约

映射可归约是一种简单的从一个问题归约到另一个问题的方法。核心理念是，通过一个**可计算函数**，将问题*A*转化为问题*B*的实例，然后解决问题*B*。

(这种思想在时间复杂度，空间负责度等都有体现)

定义：

如果存在可计算函数f：Σ*→Σ*，对每个w∈A，都有w∈A⇔f(w)∈B

则称语言A映射可归约到语言B，记　　，(后面记 A ≤_m B)称f为A到B的归约。

(看不懂没关系)

定理：

如果A ≤_m B 且*B*是可判定的，则*A*也是可判定的。

推论：

若A ≤_m B 且*A*不可判定，则*B*也是不可判定的。

(这让我想起了数列的比较判别法)

HALT_TM不可判定 (通过ATM)

HALT_TM={<M,w>|M为图灵机，对输入w停机}

E_TM不可判定 (通过ATM)

ETM={|M为图灵机，L(M)=∅}

REGULAR_TM不可判定 (通过ATM)

EDFA={<*A*>|*A*是一个*DFA*，且*L*(*A*)=∅}

EQ_TM不可判定 (通过ETM)

EQTM={<M1,M2>|M1,M2都是图灵机，且L(M1)=L(M2)}

格式：

Assume XXX is decidable. So suppose there's a TM R decides XXX. We can construct TM S to decide YYY（已知的不确定性的）, with operating:

S = "On the input <?> where ? is ??

1. ...
2. ...

..."

If R decides XXX, then S decides YYY.

But we know YYY is undecidable, so XXX is undecidable

线性有限自动机 LBA

线性有限自动机是可判定的

E_LBA 是不可判定的

别的

单射双射满射

概念

一、P（Polynomial Time，多项式时间）

- **定义**：P类问题是指那些可以在多项式时间内解决的问题。多项式时间意味着问题的解决时间随输入大小的增长而呈多项式增长，例如O(n3)等。
- P-type problems refer to those that can be solved in polynomial time. Polynomial time means that the solution time of a problem increases polynomial with the size of the input, such as O (n3).
- **特点**：P类问题是计算复杂性理论中相对容易的一类问题，因为存在已知的高效算法可以在多项式时间内找到问题的解。

二、NP（Nondeterministic Polynomial Time，非确定性多项式时间）

- **定义**：NP类问题是指那些可以在多项式时间内验证解的正确性的问题。这并不意味着NP类问题本身可以在多项式时间内找到解，但如果给出了一个解，那么可以在多项式时间内验证这个解是否正确。
- NP class problems refer to those that can verify the correctness of solutions in polynomial time. This does not mean that NP class problems themselves can find solutions in polynomial time, but if a solution is given, it can be verified in polynomial time whether the solution is correct.
- **特点**：NP类问题包含了P类问题，但可能还包括一些更难的问题。至今，人们尚未确定是否存在一个多项式时间的算法可以解决所有的NP类问题。

三、PSPACE（Polynomial Space，多项式空间）

- **定义**：PSPACE类问题是指那些可以在多项式空间内解决的问题。这里的“空间”指的是解决问题所需的内存或存储量。
- PSPACE problems refer to those that can be solved in polynomial space. The term 'space' here refers to the amount of memory or storage required to solve a problem.
- **特点**：PSPACE类问题关注的是解决问题所需的空间复杂度，而不是时间复杂度。与P类问题不同，PSPACE类问题可能包括一些在多项式时间内无法解决但在多项式空间内可以解决的问题。

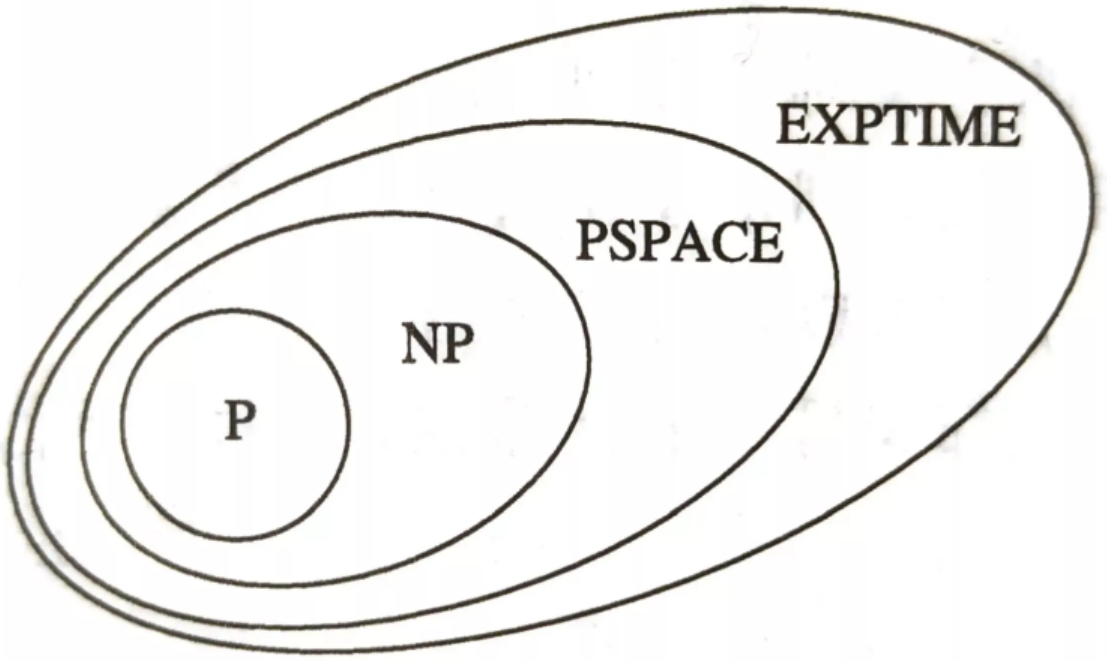
四、NPSPACE（Nondeterministic Polynomial Space，非确定性多项式空间）

- **定义**：NPSPACE类问题与NP类问题类似，但关注的是空间复杂度而不是时间复杂度。它是指那些可以在多项式空间内验证解的正确性的问题。

- NPSPACE problems are similar to NP problems, but focus on space complexity rather than time complexity. It refers to problems that can verify the correctness of solutions in polynomial space.
- **特点**：根据Savitch定理，NPSPACE类问题与PSPACE类问题是等价的。这意味着，任何可以在NPSPACE内解决的问题也可以在PSPACE内解决，反之亦然。

五、EXPTIME（Exponential Time，指数时间）

- **定义**：EXPTIME类问题是指那些需要指数时间才能解决的问题。这意味着问题的解决时间随输入大小的增长而呈指数增长，例如O(2^n)。
- EXPTIME problems refer to those that require exponential time to solve. This means that the problem-solving time increases exponentially with the size of the input, such as O (2 ^ n).
- **特点**：EXPTIME类问题通常被认为是计算复杂性理论中较难的一类问题。因为随着输入规模的增加，所需的时间将迅速变得巨大，这使得在实际应用中解决这类问题变得非常困难。



写Nondeterministically guess the sequence from u to v.给3分