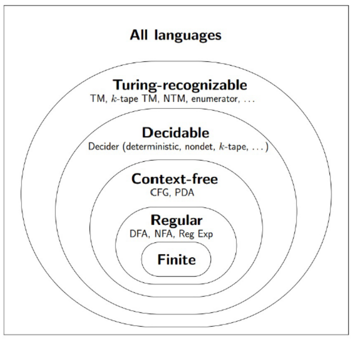
INT201_决策计算与语言-复习大纲

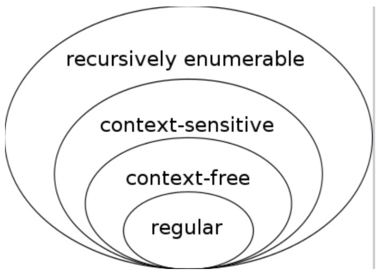
- Finite automata accept precisely the strings in the language.

  Perform a computation to determine whether a specific string is in the language.
- Regular expressions describe precisely the strings in the language

  Describe the general shape of all strings in the language.
- Context-free grammar (CFG) is an entirely different formalism for defining a class of languages.

  Give a procedure for listing off all strings in the language





Regular 正则

DFA (Deterministic Finite Automata) 确定性有限自动机

可以描述：

- 任何字符串的有限集合；
- 一些字符串的无限集合，比如：

  有且仅出现两次a的字符串；含有6个以上字母的字符串；字母b不会出现在a之前的字符串。

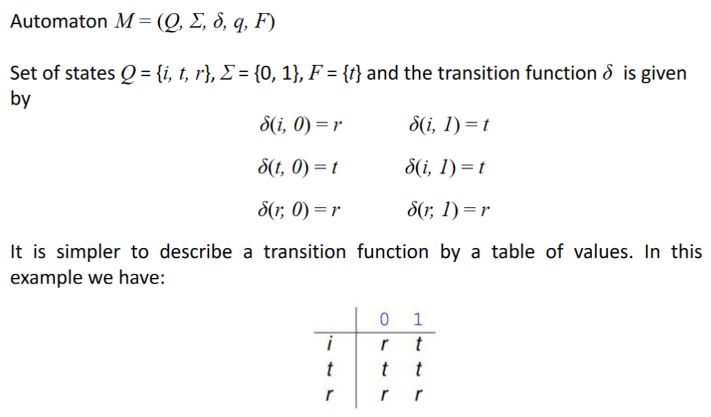
不能描述：

- 含有a的数目大于b的数目的字符串的集合；
- 回文的字符串集合；
- 格式良好的算数表达式，如果没有括号嵌套的限制（well-formed arithmetic expressions, if there is no limit on nesting of parentheses）。

定义:

五元组：$M = (Q, \Sigma, \delta, q, F)$

- Q：状态的有限集合 a finite set of states
- Σ：自动机的字母表（alphabet of the automaton），符号的有限集合（finite set of symbols）
- $\delta : Q \times \Sigma \to Q$：转移函数（transition function）
- $q \in Q$：初始态（initial state）
- $F \subseteq Q$：接受态的集合（a set of accepting/terminal states）


Automaton $M = (Q, \Sigma, \delta, q, F)$

Set of states $Q = \{i, t, r\}$, $\Sigma = \{0, 1\}$, $F = \{t\}$ and the transition function δ is given by

$$\delta(i, 0) = r \qquad \delta(i, 1) = t$$
$$\delta(t, 0) = t \qquad \delta(i, 1) = t$$
$$\delta(r; 0) = r \qquad \delta(r; 1) = r$$

It is simpler to describe a transition function by a table of values. In this example we have:

	0	1
i	*r*	*t*
t	*t*	*t*
r	*r*	*r*

DFA定义的语言:

对DFA M，word $w \in \Sigma^*$ 是accepted 或 recognized 被 M 若$\delta^*(q_0, w) \in F$，否则会被拒绝。

所有被M**接受**的words称为 the language accepted by M，记为L(M)，故：$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$

如果存在一个DFA M使得A=L(M)，那么语言A被称为是**正则的（regular）**。

正则语言对union，concatenation，Kleene star操作封闭。

NFA (Nondeterministic Finite Automata) 非确定性有限自动机

定义:

五元组：$M = (Q, \Sigma, \delta, q, F)$

- Q：状态的有限集合 a finite set of states
- Σ：自动机的字母表（alphabet of the automaton），符号的有限集合（finite set of symbols）

- $\delta : Q \times \Sigma_\epsilon \to P(Q)$：转移函数（transition function）

  > 对任意alphabet Σ，定义$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$
  >
  > 有幂集（power set）$P(Q) = \{R : R \subseteq Q\}$

- $q \in Q$：初始态（initial state）
- $F \subseteq Q$：接受态的集合（a set of accepting/terminal states）

NFA接受的语言：

$L(M) = \{w \in \Sigma^* : M\ accepts\ w\}$

DFA表示的范围和NFA等效

NFA转DFA：

子集构造法

正则语言（Regular Language）

定义：一个语言是正则的当且仅当有NFA识别他。A language is regular if and only if some NFA recognizes it.

正则语言在并集∪ union，连接concatenation，克林闭包Kleene star，补集complement，交集∩ interaction下是封闭的。

优先级：括号》*》连接》并

正式定义：

对正则表达R，L(R)记为R产生的语言。

- ϵ，$\emptyset$都是正则表达式（regular expression），描述了语言$\{\epsilon\}$，$\emptyset$
- 对每个$a \in \Sigma$，a是正则表达式，描述了语言$\{a\}$
- 若R_1, R_2是正则表达式，则$R_1 \cup R_2$，R_1R_2，R_1^*都是正则表达式，描述了语言$L_1 \cup L_2$，L_1L_2，L_1^*。

Kleene's Theorem：语言L是正则的当且仅当存在正则表达式表示L

ε-NFA（NFA with ε-moves）：允许状态之间通过 ε 转移（不消耗输入符号）。

Generalized NFA (GNFA)：一种扩展形式的 NFA，它允许状态之间的转移被正则表达式标记。

DFA转Re：

递归地把$M = (Q, \Sigma, \delta, q, F)$转为等价的GNFA G：引入起始态s终态t，将转移条件变为re；重复前一句操作直到仅剩始态s和终态t，中间即为re。

Pumping Lemma 泵引理：

用于证明一个语言不是正则语言（或不是上下文无关语言）；但不能证明一个语言属于某个语言类（因为存在满足pumping lemma的非正则语言）。

对正则语言A，存在重复长度（pumping length）$p \geq 1$，对A中每个字符串$s = xyz$（$|s| \geq p$），有：

- $y \neq \epsilon$
- $|xy| \leq p$
- for all $i \geq 0, xy^iz \in A$

通过反证法证明。

Context-Free 上下文无关

CFG (Context-Free Grammar)上下文无关文法

定义：

四元组$G = \{V, \Sigma, R, S\}$

- V：变元（variables）的有限集
- Σ：终结符（terminals）的有限集，不得与变元相同
- S：起始变元，V中的一个元素。
- R：规则（rules）的有限集 $A \rightarrow w, where\ A \in V,\ w \in (V \cup \Sigma)^*$

归约（yeild、reduction）：从字符串w到文法变元L的分析过程

派生（derive、derivation）：从文法变元到字符串的分析过程

TODO：任何正则语言都是Context Free 的证明。

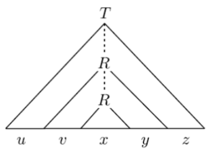
封闭性：对并，连接，Kleene star封闭。

CFL的Pumping Lemma

对CFL L，存在pumping length $p \geq 1$，满足以下条件：所有L中的字符串s，$|s| \geq p$，可写作$s = uvxyz$，满足：

- $|vy| \geq 1$（即v和y非空）
- $|vxy| \leq p$
- $uv^ixy^iz \in L, for\ all\ i \geq 0$

"Since the grammar is finite, if every variable only appear once in the parse tree, only strings of finite length can be generated. For long enough strings, there must exists variables being used at least twice."



CNF (Chomsky Normal Form) 乔姆斯基范式

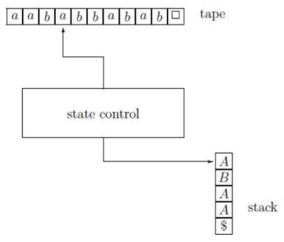
若CFG满足以下形式，则认为属于CNF：

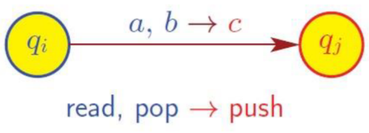
对每个R都符合以下任意一个要求：

- $A \to BC$，A，B，C都是变元，其中B和C不是起始变元；
- $A \to a$，A是变元，a是终结符；
- $S \to \epsilon$，S是起始变元。

TODO：CFL转CNF

PDA (Pushdown Automata) 下推自动机





PDA含有一个纸带，一个栈，一个状态控制：

- Tape：用于储存属于$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$的cells
- Tape head：在纸带上移动，一次向右一格。
- Stack：包含来自有限集Γ的符号，称为stack alphabet。包含$\$$，栈底符号（bottom of stack）。
- Stack head：读取栈顶符号，可以弹栈或将Γ的符号压栈。
- State control：可以处于状态（有限数量）中的任意一个。状态的集合是Q，其中有q是起始态（start state）。

在状态q_i，读纸带a，弹栈b，进入状态q_j，压栈c。

定义:

六元组$M = (Q, \Sigma, \Gamma, \delta, q, F)$

- Q：状态的有限集
- Σ：（有限的）input/ tape alphabet
- Γ：（有限的）stack alphabet
- δ：转移函数$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to P(Q \times \Gamma_\epsilon)$
- q：起始态，$\in Q$
- F：接受态的集合，$F \subseteq Q$，只要在F就行，不管栈的情况。

PDA的不确定性（Nondeterministic）：PDA转移函数允许非确定性

- PDA是非确定的，所以他不等价于DPDA（deterministic PDA）（应当是大于）

所有被PDA M接受的字符串 is the language recognized by M，记为L(M)。

> PDA和CFL是等效的
>
> If A=L(G) for some CFG G, then A=L(M) for some PDA M
>
> idea: Given G, convert it into PDA M with L(M)=L(G) by building PDA that simulates a leftmost derivation.

TODO：CFG转PDA

Recursively-Enumerable 递归可枚举

TM (Turing Machine)图灵机

满足：Regular，Context-free，Context-sensitive，recursively enumerable.

定义:

七元组$M = \{\Sigma, \Gamma, Q, \delta, q, q_{accept}, q_{reject}\}$

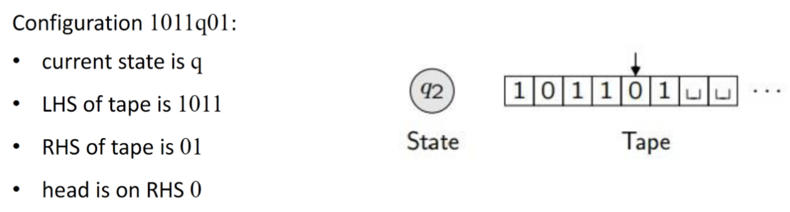
- Σ：input alphabet，有限集，不包含符号“_”
- Γ：tape alphabet，有限集，包含符号“_”（blank symbol），$\Sigma \subseteq \Gamma$
- Q：状态的集合，有限集
- q：起始态（start state），Q中的一个元素
- q_{accept}：接受态（accept state），Q中的一个元素

- q_{reject}：拒绝态（reject state），Q中的一个元素
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{Left, Right, None\}$：转移函数

在开始计算前，TM处于状态r，磁头在一个特定的cell上（起始在最左）

根据r和磁带上读到的符号k：转移到状态r'；磁头读当前符号；磁头动作。计算将持续直到进入接受态或拒绝态，不然TM会一直运行。

TM Configuration


Configuration $1011q01$:

- current state is q
- LHS of tape is 1011
- RHS of tape is 01
- head is on RHS 0

Configuration of a TM is string uqv, $u, v \in \Gamma^*$, $q \in Q$; M is in state q, tape contains uv, tape head pointing to the cell containing the first symbol in v.

Computation:

given TM M, input string $w \in \Sigma^*$.

If exist a finite sequence of configurations$C_1, C_2, ..., C_k$ for some $k \geq 1$ with:

- C_1 is the **starting configuration** $q_0 w$
- C_i **yields** C_{i+1} for all $i = 1, ..., k-1$
- C_k is an **accepting configuration** $uq_{accept}v$ for some $u, v \in \Gamma^*$

Recognizable Languages

定义:

能被图灵机M接受的语言$L(M)$是在Σ^*中所有被M接受的字符串。

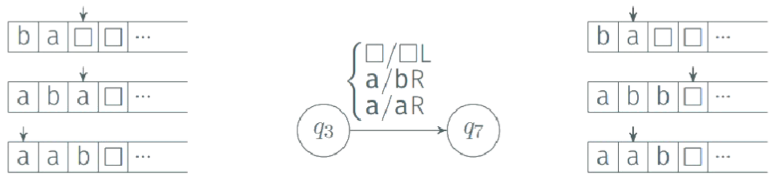
如果存在TM M满足$A = L(M)$，我们称语言A是图灵可识别的（Turing-recognizable)

- 对输入$w \notin L(M)$，图灵机M可能在拒绝态停止，或无限循环。
- 对输入$w \in L(M)$，图灵机M会在接受态停止。

可见，图灵可识别不太实用，因为我们不知道图灵机是否会停止。

Multi-tape TM

拥有k个不同的磁带和磁头；$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{Left, Right, None\}^k$：转移函数

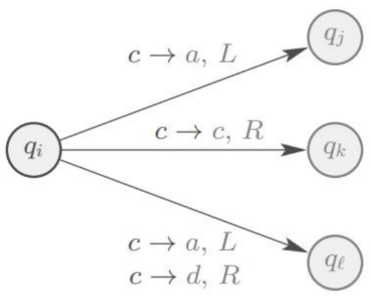


任何k-tape TM（k≥1）都可以转换为等效的Single-tape TM。可用Single-tape TM simulate k-tape TM。

NTM (Nondeterministic TM)

NTM can have several options at every step.

$\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{Left, Right\})$：转移函数



$\delta(q_i, c) = \{ (q_j, a, L), (q_k, c, R), (q_l, a, L), (q_l, d, R) \}$

With any input w, computation of NTM is represented by a configuration tree. 任意一个叶接受就接受。

任何NTM都有等效的TM

Decidable Languages

判别器（decider）是可以对任何输入停下的图灵机，即永远不会无限循环。

若对每个可能的输入$w \in \Sigma^*$ 图灵机M总是在halting configuration结束，则称语言A=L(M)是被图灵机M可判别的（decided）。

- M ends in q_{accept} for each $w \in A$
- M ends in q_{reject} for each $w \notin A$

A is **Turing-decidable** if ∃ TM M that decides A

TM for Turing-recognizable language may loop on strings w ∉ this language

- A is Turing-recognizable if A=L(M) for some TM M
- A is Turing-decidable if A=L(M) for some TM M (decider)that halts on all inputs
- Multi-tape TM and Nondeterministic TM are equivalent to TM

Uncountable Set 不可数集

定义:

An infinite set is uncountable if there is no bijection between this infinite set and the set of natural numbers (N).

即，若一个集合的势大于自然数集，则该集合是不可数集。

eg：The set R of all real numbers is uncountable

Proof idea（**Diagonalization method**）： show no mapping can be surjective from N to R

- Show the set $\{(i,j)|i,j \in N\}$ is countable.
- Show the Kleene star of any countable set of strings of finite length is countable.
- Show for any finite set S, $|P(S)| > |S|$, where$P(S)$ is the power set.

Non Turing-recognizable 非图灵可识别

The Church–Turing Thesis：图灵机的数量是countable的，但语言的数量是uncountable的。故存在语言无法被图灵机识别。

Decidability 可判定性

对语言L，其元素是(B,w)，其中B是计算模型（如DFA等），w是alphabet Σ上的字符串。

当且仅当$w \in L(B)$时，$(B,w) \in L$

Acceptance problem

Decision problem：给定模型是否接受/产生给定字符串w?

- Instance：$< B,w >$是对(B,w)的解码
  - If B="TuringMachine1"and w="1101". Then ⟨B,w⟩="TuringMachine1|1101"
- Universe：Ω 包含所有可能的Instance $\Omega = \{< B,w > |B\ is\ a\ model\ ,w\ is\ a\ string\}$

- Language：包含所有正确的Instance$L = \{< B, w > | B\ is\ a\ model\ accept\ w\} \subseteq \Omega$

证明The Language L_{DFA} is decidable：Lec10p18

DFA, NFA, CFG, TM is turing-recognizable, undecidable

The Language L_{TM} is Turing-recognizable

$L_{TM} = \{< M, w >: M\ is\ a\ TM\ accepts\ string\ w\}$

- If M accepts w, then $< M, w > \in L_{TM}$
- If M does not accept w(reject/ loop), then $< M, w > \notin L_{TM}$

证明：A universal Turing machine U simulates M on w

If M accepts w, simulation will halt and accept

If M doesn't accept w (reject or loop), TM U either reject or loops.

The Language L_{TM} is undecidable

$L_{TM} = \{< M, w >: M\ is\ a\ TM\ accepts\ string\ w\}$

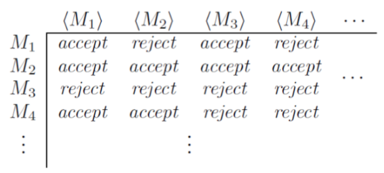
问题在于，与别的机器不同，图灵机可能永不停止，we don't really know whether the universal Turing machine will halt or not.

证明法1：通过反证法和对角化方法（diagonalization method），假设存在这样的判断器decider H，则会导致图灵机的集合是不可数的（图灵机的集合是可数的，矛盾）。

$$H(< M, w >) = \begin{cases} accept, & if\ M\ accepts\ w \\ reject, & if\ M\ does\ not\ accept\ w \end{cases}$$

Since TM is known to be countable, let's index them by M_i. So ask answer to $H(< M_i, < M_j >>)$ means whether M_i accepts the description of M_j as input.

Then we have this table:



	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	$accept$	$reject$	$accept$	$reject$	
M_2	$accept$	$accept$	$accept$	$accept$	$\cdots$
M_3	$reject$	$reject$	$reject$	$reject$	
M_4	$accept$	$accept$	$reject$	$reject$	
$\vdots$			$\vdots$		

Since H is a decider, this process will halt. Construct a Turing machine D that flips the diagonal:

- D = "On input <M>,

> 1. Run H on input<M,<M>>
>
> 2. output the opposite of what H outputs. (If H accept, reject; if H reject, accept)"

So D is a decider (also a TM), should be in the list of TM. But, by the diagonal flipping, it is not on the list as at least one value
differs, and D is not in this list. So there will be uncountable TM. This is a Contradiction.

证明法2：

If D accepts <D>, in step1 H accepts <D,<D>>, then in step2 D rejects the <D>.
If D rejects <D>, in step1 H rejects <D,<D>>, then in step2 D accepts the <D>.

We have a contradiction.

A language A is decidable if and only if it is Turing-recognizable and co-Turing-recognizable.

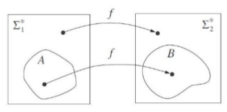
The complement of $L_{TM}(\overline{L_{TM}})$ is not Turing-recognizable.

Reducibility 归约

归约是一种将一个问题转换为另一个问题的方法，以便可以使用第二个问题的解决方案来解决第一个问题。

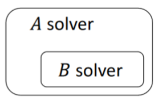
如果 A 归约到 B，则 B 的任何解都解决 A（简化总是涉及两个问题，A 和 B），A 不能比 B 更难（B可判定则A可判定，A不可判定则B不可判定）。

Mapping Reducibility of A to B:
Translate A-questions to B-questions.



w ∈ A ⇐⇒ f(w) ∈ B

(General) Reducibility of A to B:
Use B solver to solve A.



However, we have other options such as A accepts when B rejects, calling B multiple times or computing while conditioning on the B solution.

If $A \leq_m B$ and B is decidable, then A is decidable.

If $A \leq_m B$ and A is undecidable, then B is undecidable.

（可换成Turing-recognizable）

Halting problem for TMs is undecidable

LTM (acceptance problem for TMs) is undecidable, $L_{TM} = \{< M, w >: M\ is\ a\ TM\ accepts\ string\ w\}$

Define related problem: $HALT_{TM} = \{< M, w >: M \ is \ a \ TM, \ M \ halts \ on \ string \ w\}$

L_{TM} and $HALT_{TM}$ has same universe: $\Omega = \{< M, w > | M \ is \ TM, \ w \ is \ string\}$

given$< M, w > \in \Omega$:

- if M halts on input w, then M, $w \in HALT_{TM}$,
- if M doesn't halts on input w, then M, $w \notin HALT_{TM}$,

TODO: proof

Rice's Theorem

Define T to be the set of all Turing machines descriptions, i.e.,

$T = \{< M >: M \ is \ a \ TM\}$

令P是T的子集，满足：

- $P \neq \emptyset$ i.e., there exists a TM M such that $< M > \in P$,
- P is a proper subset of T i.e. there exists a TM N such that N ∉ P
- For any two TM M1 and M2 with $L(M1) = L(M2)$,

  > (a) either both M1 and M2 are in P or
  >
  > (b) none of M1 and M2 is in P.

Then the language P is undecidable.

> In general, a property is about the language, not about the TMs.

应用:

Proof:

Let P be a subset of T such that ...

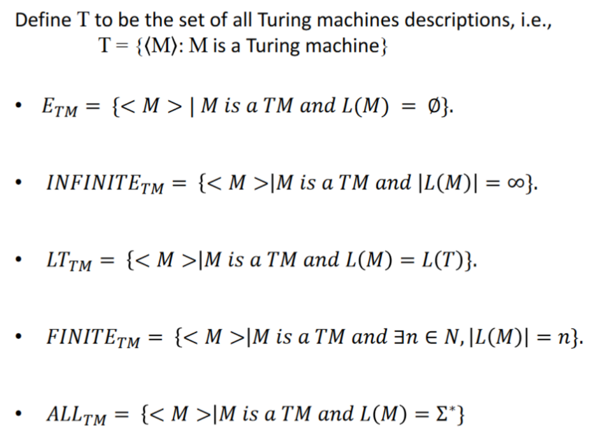
$E_{TM} = \{< M > | M \ is \ a \ TM \ and \ L(M) = \emptyset\}$ is undecidable

- a TM rejects all inputs will suffice is in the set.
- a TM accepts all unputs is not in the set.
- If $L(M1) = L(M2)$, they are both either $\emptyset$ or non-empty, wither both in the set or

  > not in the set respectively.

Therefore E_{TM} is undecidable by Rice's theorem.

同理，以下都是不可判定问题：



Define T to be the set of all Turing machines descriptions, i.e.,
T = {⟨M⟩: M is a Turing machine}

- $E_{TM} = \{< M > \mid M \text{ is a } TM \text{ and } L(M) = \emptyset\}.$
- $INFINITE_{TM} = \{< M > | M \text{ is a } TM \text{ and } |L(M)| = \infty\}.$
- $LT_{TM} = \{< M > | M \text{ is a } TM \text{ and } L(M) = L(T)\}.$
- $FINITE_{TM} = \{< M > | M \text{ is a } TM \text{ and } \exists n \in N, |L(M)| = n\}.$
- $ALL_{TM} = \{< M > | M \text{ is a } TM \text{ and } L(M) = \Sigma^*\}$