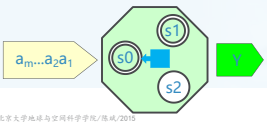
离散数学：形式语言与自动机：有限状态机

陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

句子识别

- 给定一个字符串，判定是否属于给定语法G的语言L(G)
- 一般来说，这是个难解的问题
- 对于**正则语法**和**正则语言**，可以通过一个“识别机器”来判定



机器machine

- 一个系统，接受**输入**，改变自身的**状态**，产生**输出**
- 状态指为了完成机器的任务而对输入序列进行的一种临时归类
- 可以理解为一种对记忆的模拟
- 在逐个接受输入字符的过程中，机器状态会发生**多次**改变，最终会**停止**在某个状态，并有输出**产生**
- 如果对于所有的输入，机器状态的数目有限，称为**有限状态机**（Finite State Machine）

有限状态机

- 有限状态机是一个五元组$M(A,S,Y,s_0,F)$
- A：**输入**字符串的**字母**表
- S：机器的有限**状态**集合
- $Y \subseteq S$：被称作“**接受**”的一些状态
- $s_0 \in S$：**初始**状态
- $F:S \times A \to S$：**状态转移函数**，指明在某个状态下接受输入字符所引起的状态变迁

有限状态机例子

- $A=\{a,b\}$
- $S=\{s_0,s_1,s_2\}$，$Y=\{s_0,s_1\}$，s_0初始状态
- F如下表：

F	a	b
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_2	s_2

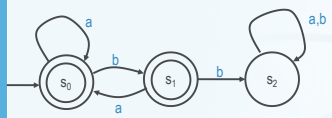
离散数学：形式语言与自动机：状态图

陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

状态图

- FSM通常用状态图来表示
- 状态图D=D(M)是边带**标记**的**有向图**
- 节点为状态，转移函数决定边的走向和标记
- 接受状态用双圈表示
- 边的定义：如果$F(s_j,a)=s_k$，则从节点s_j做标记为a的有向边到节点s_k
- 初始状态s0用一个特殊的无源箭头标识

状态图例子：赋权有向图

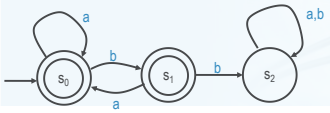


机器识别的语言

- M所识别的A上的所有字符串
- $w=a_1a_2...a_m \in A^*$，
- w确定了一个状态序列$s_0,s_1,s_2...s_m$
- 其中$F(s_{i-1},a_i)=s_i$
- 也就是确定拟路径$P=(s_0,a_1,s_1,a_2,s_2,...,a_m,s_m)$
- 如果$s_m \in Y$则称M识别w

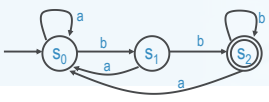
从状态图看识别语言

- 看看下图的例子是否识别：
- **ababba**、**baab**、**空串**
- 例子机器识别什么模式的字符串？
- 所有**不**包含**连续两个b**的字符串



正则语言和有限状态机

- Kleen定理：字母表A上的形式语言L是正则的当且仅当存在一个有限状态机M使得L=L(M)
- 例：构造自动机M，识别恰好以bb结尾的字符串
- 正则表达式：(a|b)*bb

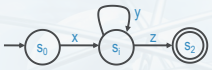


离散数学：形式语言与自动机：泵引理

陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

泵Pumping引理

- 长度**超过**状态数目的接受串w，都可以表示成w=xyz形式，而xy^nz都会被M接受
- 设w对应的状态序列为$s_0,s_1,s_2,\ldots,s_n$
- n大于状态的数目|S|，所以根据鸽笼原理必有$s_i=s_j(i<j)$
- 令$x=a_1a_2\ldots a_i$，$y=a_{i+1}\ldots a_j$，$z=a_{j+1}\ldots a_n$
- x以s_i状态结尾，xy以$s_j=s_i$状态结尾，所以xy^m也以状态s_i结尾，xy^mz结尾状态仍为s_n



证明：$L=\{a^mb^m:m>0\}$不是正则语言

- 假设L是正则语言，则有有限状态机M接受L，假设M状态个数为**k**
- 令$w=a^kb^k$，则一定有$|w|>k$，根据泵引理，$w=xyz$，其中y非空，而且xy^2z也被M接受
- 如果y**仅含有a或仅含有b**，那么xy^2z中a,b的个数就**不相等**了，应该不属于L
- 如果y**同时含有a和b，**那么y^2中一定会出现**a在b之后**的情况，xy^2z也应该不属于L
- 两个矛盾说明假设错误，L不是正则语言
- 我们以前的例子a^mcb^m也不是正则语言

离散数学：形式语言与自动机：机器同余

陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

机器同余(machine congruence)

- 机器同余是一个**等价关系**R
- 在有限状态机M(A,S,F)的**状态集合S上**的等价关系R
- 对于任意的s,t∈S，如果sRt，
- 能导出对于任意的输入符号**x∈A**，都有F(x,s)RF(x,t)
- 即：同一个等价类的状态对于任意的输入都转移到属于同一个等价类的状态

回顾抽象代数中的同余关系

- 在代数结构<S,Δ>中，S上的一个等价关系~，如果满足：
- a~b蕴涵Δa~Δb，称~是S上关于一元运算Δ的**同余关系**
- 同余关系下，载体元素体现出一种聚类性质，同一类的元素经过运算，其结果还是同一类
- 而在有限状态机中的机器同余则表现了状态之间的一种聚类性质，同一类状态经过读入字符，其目标状态还是同一类
- 同余反映了某种等效性

离散数学：形式语言与自动机：商机器

陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

商机器（quotient machine）

- 相对于有限状态机M(A,S,F)和M上的机器同余R
- S'=S/R={[s]|s∈S}
- 是S关于等价关系R的商集
- F'={<[s], [F(x,s)]>｜s∈S, x∈A}
- M'(A,S',F')称为M关于R的**商机器**M/R
- 通常商机器比原来的机器简单一些，是机器化简的基础

商机器例子

- A={a,b}，S=(s_0,s_1,s_2,s_3,s_4,s_5)，F如表
- S上等价关系R的关系矩阵如图
- 对照检查F，R是机器同余

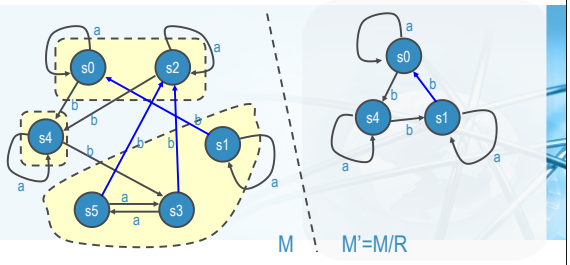
F	a	b
s_0	s_0	s_4
s_1	s_1	s_0
s_2	s_2	s_4
s_3	s_5	s_2
s_4	s_4	s_3
s_5	s_3	s_2

[s0]={s0,s2}
[s1]={s1,s3,s5}
[s4]={s4}

1	0	1	0	0	0
0	1	0	1	0	1
1	0	1	0	0	0
0	1	0	1	0	1
0	0	0	0	1	0
0	1	0	1	0	1

F'	a	b
$[s_0]$	$[s_0]$	$[s_4]$
$[s_1]$	[s1]	$[s_0]$
$[s_4]$	$[s_4]$	$[s_1]$

商机器例子：状态图



离散数学：形式语言与自动机：商机器性质

陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

商机器应用

- 有限状态机可能出现**冗余**的状态
- 不同的机器可能接受相同的语言
- 其中状态数目最少的机器为最简
- 通过构造**机器同余**及商机器来合并这些冗余状态，**减少**状态数目

对字符串相容的状态

- s,t∈S，w∈A*
- 如果F(s,w)∈Y并且F(t,w)∈Y
- 或者F(s,w)∉Y并且F(t,w)∉Y
- **注**：F(s,w)=F(..F(F(F(s,a1),a2),a3...),an)
- 称s,t是关于w**相容**的
- 定义：S上的二元关系R
- 如果s,t∈S对于任何w∈A*都相容，则**sRt**
- 说明s,t其实是关于句子**识别**的**等效**状态

关于相容的二元关系R

- R是等价关系：
- 首先，具有自反、对称性
- 其次，sRt, tRu则F(s,w), F(t,w), F(u,w)要么都属于Y，要么都不属于Y，所以也有sRu，是传递的

关于相容的二元关系R

- R是机器同余：
- 要证明对于任意s,t∈S，x∈A，如果sRt，有F(s,x)RF(t,x)
- 设任意w∈A*，w'=xw
- 因为sRt，则有F(s,w')和F(t,w')同时属于或不属于Y
- 也就是F(F(s,x),w)和F(F(t,x),w)同时属于或不属于Y，即F(s,x)RF(t,x)，所以R是机器同余

商机器的性质

- 由具有相容性质的机器同余，所构造的商机器M/R
- 从初始状态开始，对任何字符串与原机器做出的是否接受结论是相同的
- M和商机器M/R识别**相同**的语言
- 由于商机器状态不多于原机器，M/R即M化简的结果
- 但如何对于长度不限制的任意w找到这样的等价关系R？

离散数学：形式语言与自动机：机器化简

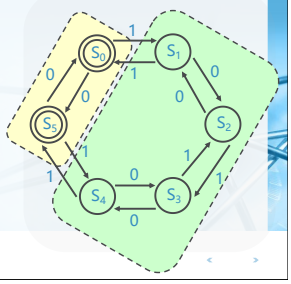
陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

相容关系R的迭代算法

- S的初始划分P_0={Y, Y~}
- 假设目前已有划分$P_k=\{S_1,S_2,\ldots S_m\}$，考察每个等价类$S_i$
- 如果S_i中的两个状态s,t在**所有**的输入x作用下都转移到**同一个**状态分块A_j（取决于x），则据此构造S_i的进一步细分
- 将所有S_i的细分合在一起，成为新的划分P_{k+1}
- 如果$P_{k+1}=P_k$，算法停止，否则回到第2步
- 根据R构造商机器M/R即为化简结果

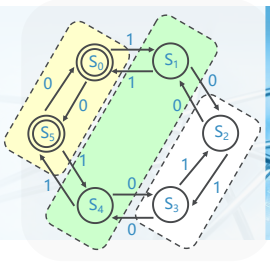
机器化简的例子

- 从$P_0=\{\{s_0,s_5\},\{s_1,s_2,s_3,s_4\}\}$开始
- 首先先看：{s_0,s_5}
- 对0：转到{s_0,s_5}；对1：转到{s_1,s_4}
- 不分裂。
- 再看{s_1,s_2,s_3,s_4}
- 对0：s_1,s_2,s_3,s_4都转到Y~
- 对1：s_1,s_4转到Y，s_2,s_3转到Y~
- 分裂为{s_1,s_4},{s_2,s_3}



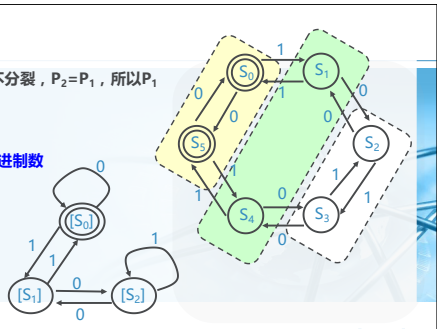
机器化简的例子

- P1={{s_0,s_5}, {s_1,s_4}, {s_2,s_3}}
- 看{s_0,s_5}：
- 对0转到{s_0,s_5}；对1转到{s_1,s_4}；不分裂；
- 看{s_1,s_4}：
- 对0转到{s_2,s_3}；对1转到{s_0,s_5}；不分裂
- 看{s_2,s_3}：
- 对0转到{s_1,s_4}；对1转到{s_2,s_3}；不分裂



化简的机器

- P_1中所有的分块都不分裂，P_2=P_1，所以P_1对应机器同余R
- 商机器M/R
- 识别能被**3整除**的**二进制数**
- [s_0]被3整除的状态
- [s_1]余1的状态
- [s_2]余2的状态



离散数学：形式语言与自动机：带输出的机器

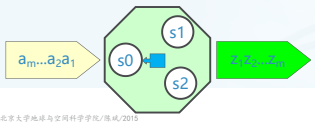
陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

带输出的机器

- 在有限状态机M(A,S,Y,s_0,F)的基础上
- 去掉接受状态集合Y，增加：
- **输出字符**的有限集合Z
- **输出函数**G:S×A→Z
- 在状态转移的同时，输出一个字符

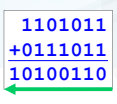
带输出机器的工作方式

- 输入字符串u=$a_1a_2…a_m$印在一条输入带上
- 机器M逐个读入输入带的字符
- 进行状态转移V=$s_0s_1s_2…s_m$
- 同时在输出带上逐个打印输出字符：
- w=$z_1z_2…z_m$



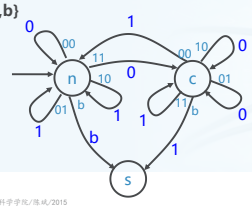
实现二进制加法的有限状态机

- 两个二进制数的加法：
- 通过给较短的数前面补0，使两个数具有相同的长度；
- 将两个数**对应位**组合，并从低位开始输入到机器，输入特定的“**空白**”表示**输入结束**
- 输入：11,11,00,11,01,11,10,**b**
- 输出：0,1,1,0,0,1,0,1
- 计算结束，进入“**停机**”状态



实现二进制加法的有限状态机

- M=(A,S,Z,s0,F,G)，A={00,01,10,11,b}
- S={执行(c)，初始(n)，停止(s)}
- Z={0,1,b}



离散数学：形式语言与自动机：程序实现状态机

陈斌 北京大学地球与空间科学学院 gischen@pku.edu.cn

用程序实现有限状态机

```
status=s0;
while(a=get_next_input())
switch (status) of {
  s0: switch(a) of {
    'x' : status= s3; output( '0' ); break;
    'y' : status= s4; output( '1' ); break;
  }; break;
  s1:..
};
return status;
```


有限状态机的限制

- 不存在能作“操作数位数无限制”的二进制乘法的有限状态机
- 现实中的计算机，内存有限，能够表达的状态有限，属于一种有限状态机
- 能够做位数有限制的乘法
- 如果宇宙是有限的，理论上也是有限状态机
- 经过足够长的时间，总会回到以前的某个状态，再重演历史
- 是循环宇宙论的主要论据☺