2.4.3 最小化DFA

正规式—NFA—DFA—最小DFA

引入一个“可区分”的概念：

定义2.7 对于DFA中的任何两个状态t和s，若从一状态出发接受输入字符串ω，而从另一状态出发不接受ω，则称ω区分状态t和s。如果存在某个能够区分状态s和状态t的串，那么它们就是可区分的。 ■

反方向思考定义2.7：

设想任何输入序列ω对s和t均是不可区分的，则说明从s出发和从t出发，分析任何输入序列ω均得到相同结果。

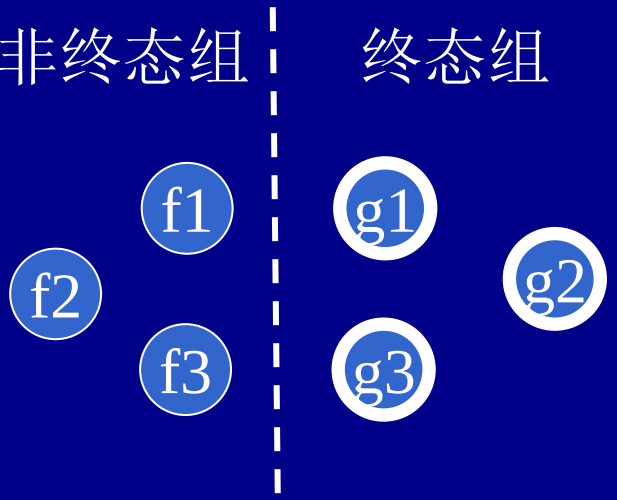
因此，s和t可以合并成一个状态。

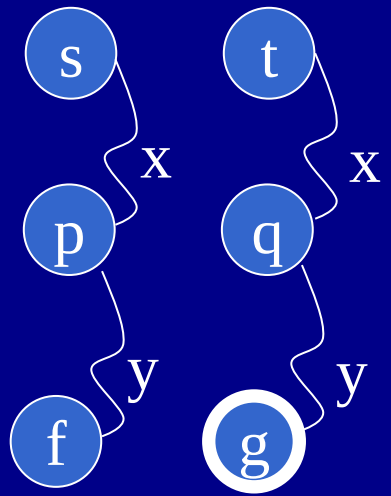
最小化DFA的本质：

将DFA中的状态分成不同的组，使得同一组中的状态之间不可区分，不同组的状态之间是可区分的。

给定一个DFA,

- DFA的终态和非终态是可区分的；
- 如果分别从s和t出发，沿着标记为x的路径到达的两个状态是可区分的，则s和t也是可区分的。





基于已区分的非终态组和终态组，进一步分裂非终态组。
例如，若对于同一字符a，move(f2,a)=f1 move(f3,a)=g1，因为f1和g1可区分，所以f2和f3可区分。

终态组的分裂方法类似。

算法2.6 最小化DFA的状态数

输入 DFA D={S，∑，move，s0，F}。

输出 等价的D'={S'，∑，move'，s0'，F'}（D'状态数最少）

方法 执行如下步骤：

1. 初始划分Π={S-F，F}；Πnew =Π。F是终态集， S-F是非终态集
2. 应用下述过程构造新的划分Πnew：

   for Π的每一个组G

   loop 划分G，G的两个状态s和t在同一组中的充要条件是：

   $\forall a \in \sum \forall Gi \in \Pi .(move(s,a) \in Gi \leftrightarrow move(t,a) \in Gi)$

   用新划分的组替代G，形成新的划分Πnew；

   end loop；
3. 若Πnew=Π，令Πfinal=Π，转4；否则令Π=Πnew并重复步骤2；

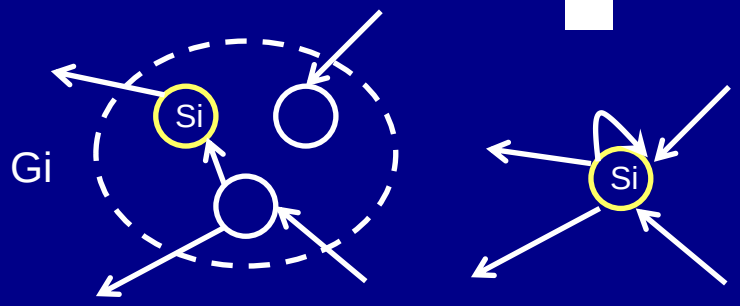
对充要条件的解释：对于任意字母表字符a，
move(s,a)和move(t,a)都在Π的相同分组中，或者都没有下一状态。

4. **选代表并修改状态转移**：在Πfinal每个组Gi中选一个代表si'，使得D中从Gi所有状态出发的状态转移在D'中均从si'出发，D中所有转向Gi中的状态转移在D'中均转向si'；

   初态：含有D中s0的状态组G0的代表s0'，称为D'的初态；

   终态：D中所有含F中状态的Gj的代表sj'构成D'的终态集F'；

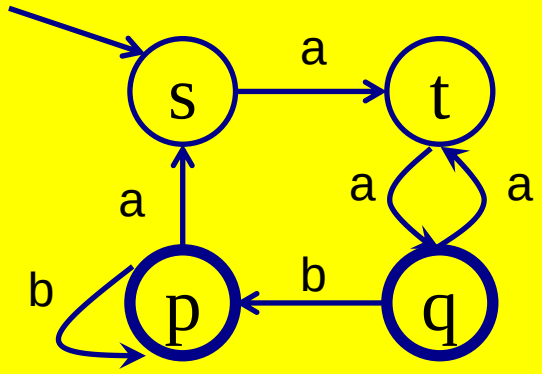
5. 删除死状态，即不是终态且对所有输入字符均转向其自身，或从初态不可到达的状态。 ■



最小化DFA算法的主要步骤

1. 初始划分：终态与非终态；
2. 利用可区分的概念，反复分裂划分中的组Gi，直到不可再分裂；
3. 由最终划分构造D'，关键是选代表和修改状态转移；
4. 消除可能的死状态和不可达状态。

DFA D=(S, ∑, move, s, F)



2、Π={G11={s},G12={t}, G2={p, q}}

G11和G12都只含一个状态，所以不会再分裂；

G2:

move(p,a)=s∈G11
move(q,a)=t∈G12
} a可以区分p和q，分裂G2为G21和G22

move(p,b)=p∈G2
move(q,b)=p∈G2
} b不能区分p和q

Π_{new}={G11={s},G12={t},G21={p},G22={q}}

1、Π={G1={s, t},G2={p, q}}

G1:

move(s,a)=t∈G1
move(t,a)=q∈G2
} a可以区分s和t，分裂G1为G11和G12

G2:

move(p,a)=s∈G1
move(q,a)=t∈G1
} a不能区分p和q

move(p,b)=p∈G2
move(q,b)=p∈G2
} b不能区分p和q

Π_{new}={G11={s},G12={t},G2={p, q}}

3、Π={G11={s},G12={t}, G21={p}, G22={q}}

每个分组都只含一个状态，所以不会再分裂；

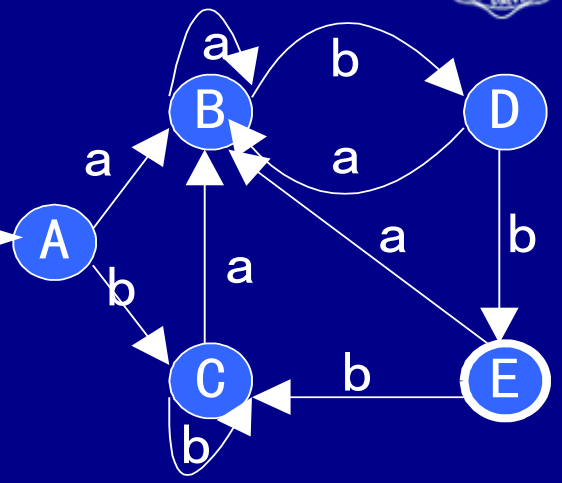
Π_{new}={G11={s},G12={t},G21={p}, G22={q}}

= Π

= Π_{final}

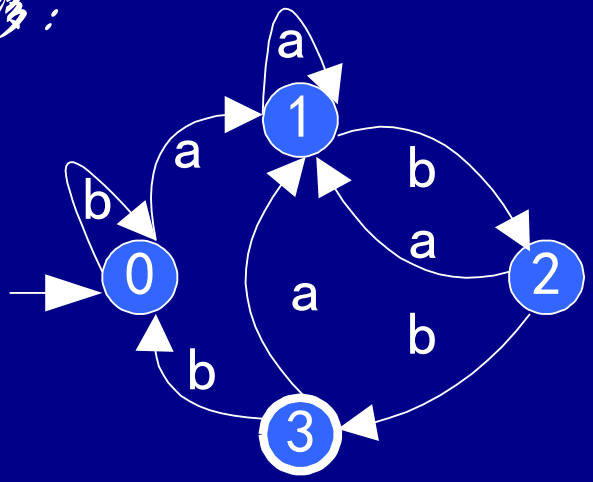
例2.17 用算法2.6化简DFA

1. 初始化划分 Π1={ABCD, E}
2. 根据算法中步骤2，反复分裂划分中的组：
   - ① ∵ m(D, b)=E ∴ Π2={ABC, D, E}
   - ② ∵ m(B, b)=D ∴ Π3={AC, B, D, E}
   - ③ Π3? 于是：Πfinal={AC, B, D, E}



4. 根据Πfinal构造D'：
   - ① 选代表，用A代表AC组；
   - ② 修改状态转移：

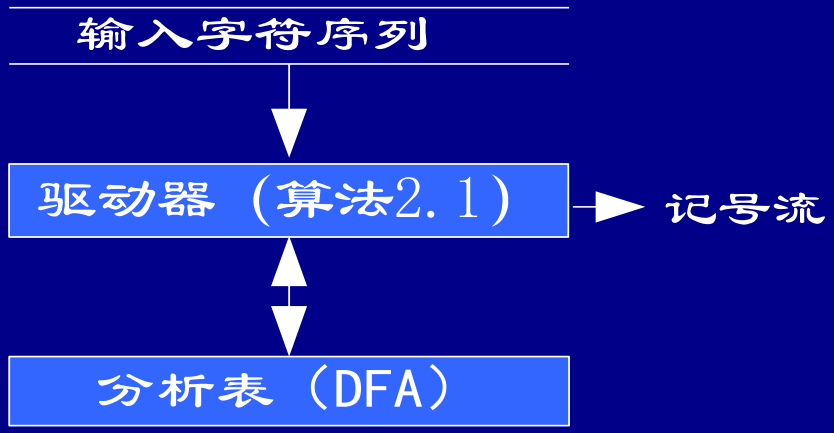
用0、1、2、3代替A、B、D、E：



m(A, a)=B, m(A, b)=C
m(B, a)=B, m(B, b)=D
m(C, a)=B, m(C, b)=C
m(D, a)=B, m(D, b)=E
m(E, a)=B, m(E, b)=C
m(A, a)=B, m(A, b)=A
m(B, a)=B, m(B, b)=D
m(D, a)=B, m(D, b)=E
m(E, a)=B, m(E, b)=A

2.4.4 由DFA构造词法分析器

<1> 表驱动型的词法分析器



其中，需要：

1. 有适当的数据结构存放DFA；
2. 修改算法2.1，以适应实际输入：
   - 识别整个文件，而不是一个记号；
   - 满足最长匹配原则。

对于输入序列：

result := a + b

正确的识别：id := id + id

错误的识别：

1. 仅识别一个：　　id　　(result)
2. 不满足最长匹配：id　id ...　(r或re或res ...)

<2> 直接编码的词法分析器

在表驱动的词法分析器中，DFA是被动的，需要一个驱动器来模拟DFA的行为，以实现对输入序列的分析。

直接编码的词法分析器，将DFA和DFA识别输入序列的过程合并在一起，直接用程序代码模拟DFA识别输入序列的过程。

问题：

如何用程序模拟DFA识别输入序列的过程？即如何用程序模拟DFA的状态和它的状态转移？

1. 状态和状态转移与语句的对应关系

① 初态→程序的开始；

② 终态→程序的结束（不同终态return不同记号）；

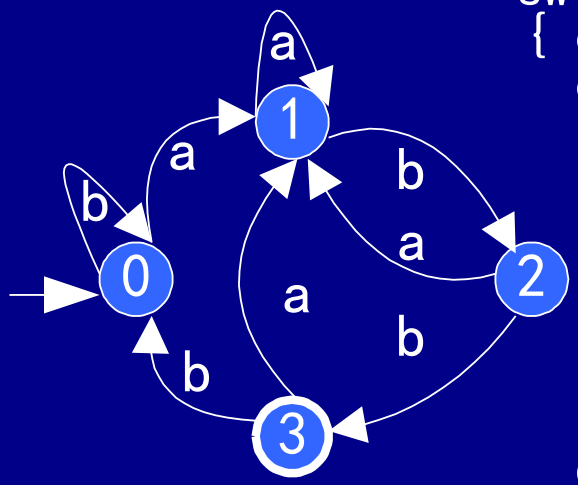
③ 状态转移→分情况或者条件语句（case/if）；

④ 环→循环语句（loop）；

⑤ return满足最长匹配原则。

2. 识别(a|b)*abb的程序框架 <2> 直接编码的词法分析器（续1）

```
void main(){  char buf[]="abba#", *ptr=buf;
  while (*ptr!='#' )
   {
I0: while (*ptr=='b') ptr++;                    // state 0
I1: while (*ptr=='a') ptr++;                    // state 1
    switch (*ptr)
    { case 'a': goto I1;
      case 'b': ptr++;
                switch (*ptr)                   // state 2
                { case 'a': goto I1;
                  case 'b': ptr++;
                            switch (*ptr)  // state3
                            { case 'a': goto I1;
                              case 'b': goto I0;
                              case '#': cout<<"yes"<<endl;
                                        return;
                              default:  break;
                            }
                  default: break;
                }
      default: break;
    }
    break;      // 遇到非法字符
  }
  cout << "no" << endl;
} // 看实例运行
```



〈3〉 两类分析器的比较

	表驱动	直接编码
分析器的速度	慢	快
程序与模式的关系	无关	有关
分析器的规模	较大	较小
适合的编写方法	工具生成	手工编写

2.4.5 词法分析器生成器简介

<1> 构造词法分析器的全过程均有算法：

正规式—NFA—DFA—最小化DFA—词法分析器（分析表）

<2> LEX的基本结构

根据正规式构造的分析表＋驱动器框架（不变的）

<3> 利用LEX构造词法分析器的关键

① 用LEX提供的正规式集合设计记号的模式；

② 用LEX提供的语义支持识别记号或指出输入中的错误。

2.5 本章小结

词法分析的两个重要环节：规定所有合法输入＋识别合法输入

重要内容：

<1> 记号、模式与单词

<2> 记号的说明：模式的形式化描述－正规式

<3> 记号的识别：有限自动机

NFA：与正规式有对应关系，易于构造，状态数少；

DFA：确定性便于记号识别，不易构造，状态数可能多；

记号识别的方法：

a. 模拟DFA；

b. 模拟NFA（特殊情况下）：需要动态计算状态子集。

<4> 从正规式到词法分析器（等价变换的过程）

- 正规式描述模式
- 由正规式构造NFA
- NFA的确定化（子集法：smove， ε-闭包）
- DFA的最小化（可区分概念）
- 词法分析器：表驱动（自动生成）与直接编码（手工编写）