

目录

- 1 Hopcroft’s algorithm 3
 - 1.0.1 algorithm 3
 - 1.0.2 Minimization example 1 3
 - 1.1 Minimization example 2 15
 - 1.2 Minimization example 3 ($1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$) 18
 - 1.3 Minimization example 4 34
 - References 39

Chapter 1

Hopcroft's algorithm

An automaton is *accessible* when for any state $p \in Q$, there exists a word $w \in V^*$ such that $q_0 \cdot w = p$. It is *co-accessible* when for any state $p \in Q$, there exists $w \in V^*$ such that $p \cdot w \in F$.

1.0.1 algorithm

Member function `min_Hopcroft` implements Hopcroft's $n \log n$ minimization algorithm, as presented in [[WATSON94b], Algorithm 4.8].

The combination of the out-transitions of all of the States is stored in a **CRSet** C .

Set L from the abstract algorithm is implemented as a mapping from States to int (an array of int is used).

Array L should be interpreted as follows: if State q a representative, then the following pairs still require processing (are still in abstract set L):

$$([q], C_0), \dots, ([q], C_{L(q)-1})$$

The remaining pairs do not require processing:

$$([q], C_{L(q)}), \dots, ([q], C_{|C|-1})$$

This implementation facilitates quick scanning of L for the next valid State-CharRange pair.

1.0.2 Minimization example 1

CRSet C; // the out labels of State's: 'a' 'b'

int L[9]; // the index of L = q: 对应等价类 [q]; L[q] 表示正在处理等价类 [q] 的字符在 C 中的 index。

$L = \{0, 0, 0, 0, 0, 0, 0, 0, 0\}$

DFA

$Q = [0, 9)$

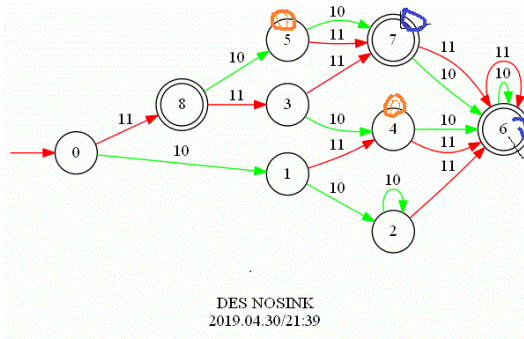
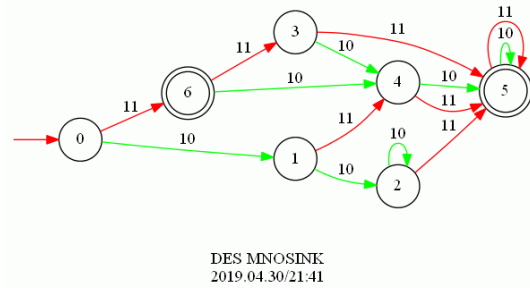
$S = \{ 0 \}$

Algorithm 1 Hopcroft's minimization algorithm**Input:** $G = (Q, V, T, q_0, F)$ **Output:** The equivalence classes of Q

```

1:  $P \leftarrow [Q]_{E_0} = \{F, Q \setminus F\}$   $\triangleright$  The initial partitions is  $[Q]_{E_0}$ , it's the total euivalence relation.
2:  $L \leftarrow \emptyset$   $\triangleright$  The waiting set
3: for all  $a \in V$  do
4:    $ADD((\min(F, Q \setminus F), a), L)$   $\triangleright$  initialization of the waiting set
5: end for
6: while  $L \neq \emptyset$  do
7:    $P_{old} = P$ ;
8:    $(Q_1, a) \leftarrow TakeSome(L)$   $\triangleright$  Take and remove some splitter
9:    $L = L \setminus \{(Q_1, a)\}$ ;
10:  for all  $Q_0 \in P_{old}$  do
11:     $Q_0$  is split by  $(Q_1, a)$   $\triangleright$  Compute the split,  $Q_0$  is splitted into  $Q'_0$  and  $Q''_0$ 
12:     $Q'_0 = \{p | p \in Q_0 \wedge T(p, a) \in Q_1\}$ 
13:     $Q''_0 = \{Q_0 \setminus Q'_0\}$ 
14:     $P = P \setminus \{Q_0\} \cup \{Q'_0, Q''_0\}$   $\triangleright$  Refine the partition, Replace  $Q_0$  by  $Q'_0$  and  $Q''_0$  in  $P$ .
15:    for all  $b \in V$  do  $\triangleright$  Update the waiting set
16:      if  $(Q_0, b) \in L$  then
17:         $L = L \setminus \{(Q_0, b)\} \cup \{(Q'_0, b), (Q''_0, b)\}$   $\triangleright$  Replace  $(Q_0, b)$  by  $(Q'_0, b)$  and  $(Q''_0, b)$  in  $L$ 
18:      else
19:         $ADD((\min(Q', Q''), b), L)$ 
20:      end if
21:    end for
22:  end for
23: end while

```

(a) DFA, Eq class= $\{\{4,5\}, \{6,7\}\}$ 

(b) mini-DFA

图 1.1: Minimization example 1

Algorithm 2 Hopcroft's minimization algorithm**Input:** $G = (Q, V, T, q_0, F)$ **Output:** The equivalence classes of Q

```

1:  $P \leftarrow [Q]_{E_0} = \{F, Q \setminus F\}$   $\triangleright$  The initial partitions is  $[Q]_{E_0}$ , it's the total euivalence relation.
2:  $L \leftarrow 0$   $\triangleright$  The waiting set
3:  $C = V$   $\triangleright$  C is all symbols set
4: if  $|F| \leq |Q \setminus F|$  then  $\triangleright$  initialization of the waiting set
5:    $L[q] = C.Size(), [q]$  is the representative of the  $F$ 
6: else
7:    $L[q] = C.Size(), [q]$  is the representative of the  $Q \setminus F$ 
8: end if
9: while (1) do
10:   if all  $L[q]=0$  then
11:     break;
12:   end if
13:   Find the first pair in L that still needs processing.  $(Q_1, a) = [q], L[q] \neq 0$   $\triangleright$  Take and remove some splitter
14:    $P_{old} = P$   $\triangleright$  current partitions
15:    $L[q] - -;$   $\triangleright$  Mark this element of L as processed.
16:   for all  $Q_0 \in P_{old}$  do
17:      $Q_0$  is split by  $(Q_1, a)$   $\triangleright$  Compute the split,  $Q_0$  is splitted into  $Q'_0$  and  $Q''_0$ 
18:      $Q'_0 = \{p | p \in Q_0 \wedge T(p, a) \in Q_1\}$ 
19:      $Q''_0 = \{Q_0 \setminus Q'_0\}$ 
20:      $P = P \setminus \{Q_0\} \cup \{Q'_0, Q''_0\}$   $\triangleright$  Refine the partition, Replace  $Q_0$  by  $Q'_0$  and  $Q''_0$  in  $P$ .
21:      $p = Q_0$ 
22:      $r = Q'_0$ 
23:     if  $[r] \neq Invalid$  then  $\triangleright$  Update the waiting set
24:       if  $([p] \leq |[r]|)$  then
25:          $L[r] = L[p]$   $\triangleright$  [r] 待处理 L[p] 剩下的字符
26:          $L[p] = C.size()$   $\triangleright$  新的 [p], 待处理 C[0]...C[C.size()-1]
27:       else
28:          $L[r] = C.size()$   $\triangleright$  // 新的 [r], 待处理 C[0]...C[C.size()-1]
29:       end if
30:     end if
31:   end for
32: end while

```

$F = \{ 6 \ 7 \ 8 \}$

Transitions =

$0 \rightarrow \{ 'a' \rightarrow 1 \quad 'b' \rightarrow 8 \}$

$1 \rightarrow \{ 'a' \rightarrow 2 \quad 'b' \rightarrow 4 \}$

$2 \rightarrow \{ 'a' \rightarrow 2 \quad 'b' \rightarrow 6 \}$

$3 \rightarrow \{ 'a' \rightarrow 4 \quad 'b' \rightarrow 7 \}$

$4 \rightarrow \{ ['a', 'b'] \rightarrow 6 \}$

$5 \rightarrow \{ ['a', 'b'] \rightarrow 7 \}$

$6 \rightarrow \{ ['a', 'b'] \rightarrow 6 \}$

$7 \rightarrow \{ ['a', 'b'] \rightarrow 6 \}$

```
8->{ 'a'->5 'b'->3 }
```

```
current = -1
```

```
is the DFA Useful?: 1
```

The combination for all the out labels of State's: $C = \{ 'a' 'b' \}$

L:

```
0 1 2 3 4 5 6 7 8
```

```
0 0 0 0 0 0 0 0 0
```

```
Initialize partitions, E0:
```

```
StateEqRel
```

```
{ 0 1 2 3 4 5 }
```

```
{ 6 7 8 }
```

```
Initialize Lrepr={F}:
```

```
{ 6 }
```

L:

```
0 1 2 3 4 5 6 7 8
```

```
0 0 0 0 0 0 2 0 0
```

```
===== Iterate: k=1
```

L:

```
0 1 2 3 4 5 6 7 8
```

```
0 0 0 0 0 0 1 0 0
```

```
Partitions:
```

```
StateEqRel
```

```
{ 0 1 2 3 4 5 }
```

```
{ 6 7 8 }
```

```
pick[q] in L: ([q], a) = ([6], 'b')
```

```
split[p] w.r.t ([6], 'b')
```

```
==split[0] w.r.t ([6], 'b')
```

```
new split of [0] is [1]
```

```
[p] = { 0 2 3 4 5 }
```

```
[r] = { 1 }
```

```
p and r are the new representatives. Now update L with the smallest of
[0] and [1]
```

```
using [r] = [1], L[r] = C.size();
```

```
after update L:
```

L:

```
0 1 2 3 4 5 6 7 8
```

```

0 2 0 0 0 0 1 0 0
==split [6] w.r.t ([6], 'b')
new split of [6] is [8]
[p] = {6 7}
[r] = {8}
p and r are the new representatives. Now update L with the smallest of
[6] and [8]
using [r] = [8], L[r] = C.size();
after update L:
L:
0 1 2 3 4 5 6 7 8
0 2 0 0 0 0 1 0 2
===== Iterate: k = 2

L:
0 1 2 3 4 5 6 7 8
0 1 0 0 0 0 1 0 2
Partitions:
StateEqRel
{0 2 3 4 5}
{1}
{6 7}
{8}

pick [q] in L: ([q], a) = ([1], 'b')
split [p] w.r.t ([1], 'b')
==split [0] w.r.t ([1], 'b')
new split of [0] is [-1]
==split [1] w.r.t ([1], 'b')
new split of [1] is [-1]
==split [6] w.r.t ([1], 'b')
new split of [6] is [-1]
==split [8] w.r.t ([1], 'b')
new split of [8] is [-1]
===== Iterate: k = 3

L:
0 1 2 3 4 5 6 7 8
0 0 0 0 0 0 1 0 2
Partitions:
StateEqRel
{0 2 3 4 5}
{1}

```

$\{ _6_7_ \}$
 $\{ _8_ \}$

$\text{pick_}[q] _ \text{in_} L : ([q], a) = ([1], 'a')$

$\text{split_}[p] _ \text{w.r.t_} ([1], 'a')$

$\equiv \text{split } [0] _ \text{w.r.t_} ([1], 'a')$

$\text{new_split_of_} [0] _ \text{is_} [2]$

$[p] = \{ _0_ \}$

$[r] = \{ _2_3_4_5_ \}$

$p _ \text{and_} r _ \text{are_the_new_representatives.} _ \text{Now_update_} L _ \text{with_the_smallest_of_}$
 $[0] _ \text{and_} [2]$

$\text{using_} [p] _ = _ [0], L[r] = L[p]; _ L[p] = C.size();$

$\text{after_update_} L :$

$L :$

$0_1_2_3_4_5_6_7_8$

$2_0_0_0_0_0_1_0_2$

$\equiv \text{split } [1] _ \text{w.r.t_} ([1], 'a')$

$\text{new_split_of_} [1] _ \text{is_} [-1]$

$\equiv \text{split } [6] _ \text{w.r.t_} ([1], 'a')$

$\text{new_split_of_} [6] _ \text{is_} [-1]$

$\equiv \text{split } [8] _ \text{w.r.t_} ([1], 'a')$

$\text{new_split_of_} [8] _ \text{is_} [-1]$

$\equiv _ \text{Iterate :_} k _ = _ 4$

$L :$

$0_1_2_3_4_5_6_7_8$

$1_0_0_0_0_0_1_0_2$

Partitions :

StateEqRel

$\{ _0_ \}$

$\{ _1_ \}$

$\{ _2_3_4_5_ \}$

$\{ _6_7_ \}$

$\{ _8_ \}$

$\text{pick_}[q] _ \text{in_} L : ([q], a) = ([0], 'b')$

$\text{split_}[p] _ \text{w.r.t_} ([0], 'b')$

$\equiv \text{split } [0] _ \text{w.r.t_} ([0], 'b')$

$\text{new_split_of_} [0] _ \text{is_} [-1]$

$\equiv \text{split } [1] _ \text{w.r.t_} ([0], 'b')$

$\text{new_split_of_} [1] _ \text{is_} [-1]$

$\equiv \text{split } [2] _ \text{w.r.t_} ([0], 'b')$


```

new_split_of [2] is [-1]
==split [6] w.r.t ([0], 'b')
new_split_of [6] is [-1]
==split [8] w.r.t ([0], 'b')
new_split_of [8] is [-1]
=====Iterate: k=5

L:
0 1 2 3 4 5 6 7 8
0 0 0 0 0 0 1 0 2
Partitions:
StateEqRel
{ 0 }
{ 1 }
{ 2 3 4 5 }
{ 6 7 }
{ 8 }

pick [q] in L: ([q], a) = ([0], 'a')
split [p] w.r.t ([0], 'a')
==split [0] w.r.t ([0], 'a')
new_split_of [0] is [-1]
==split [1] w.r.t ([0], 'a')
new_split_of [1] is [-1]
==split [2] w.r.t ([0], 'a')
new_split_of [2] is [-1]
==split [6] w.r.t ([0], 'a')
new_split_of [6] is [-1]
==split [8] w.r.t ([0], 'a')
new_split_of [8] is [-1]
=====Iterate: k=6

L:
0 1 2 3 4 5 6 7 8
0 0 0 0 0 0 0 0 2
Partitions:
StateEqRel
{ 0 }
{ 1 }
{ 2 3 4 5 }
{ 6 7 }
{ 8 }

```

```

pick [q] in L: ([q], a) = ([6], 'a')
split [p] w.r.t ([6], 'a')
==split [0] w.r.t ([6], 'a')
new split of [0] is [-1]
==split [1] w.r.t ([6], 'a')
new split of [1] is [-1]
==split [2] w.r.t ([6], 'a')
new split of [2] is [4]
[p] = {2, 3}
[r] = {4, 5}
p and r are the new representatives. Now update L with the smallest of
[2] and [4]
using [p] = [2], L[r] = L[p]; L[p] = C.size();
after update L:
L:
0 1 2 3 4 5 6 7 8
0 0 0 2 0 0 0 0 0 2
==split [6] w.r.t ([6], 'a')
new split of [6] is [-1]
==split [8] w.r.t ([6], 'a')
new split of [8] is [-1]
===== Iterate: k = 7
L:
0 1 2 3 4 5 6 7 8
0 0 0 1 0 0 0 0 0 2
Partitions:
StateEqRel
{0}
{1}
{2, 3}
{4, 5}
{6, 7}
{8}

pick [q] in L: ([q], a) = ([2], 'b')
split [p] w.r.t ([2], 'b')
==split [0] w.r.t ([2], 'b')
new split of [0] is [-1]
==split [1] w.r.t ([2], 'b')
new split of [1] is [-1]
==split [2] w.r.t ([2], 'b')

```

```

new_split_of [2] is [-1]
==split [4] w.r.t ([2], 'b')
new_split_of [4] is [-1]
==split [6] w.r.t ([2], 'b')
new_split_of [6] is [-1]
==split [8] w.r.t ([2], 'b')
new_split_of [8] is [-1]
=====Iterate: k=8

L:
0 1 2 3 4 5 6 7 8
0 0 0 0 0 0 0 0 2
Partitions:
StateEqRel
{0}
{1}
{2 3}
{4 5}
{6 7}
{8}

pick [q] in L: ([q], a) = ([2], 'a')
split [p] w.r.t ([2], 'a')
==split [0] w.r.t ([2], 'a')
new_split_of [0] is [-1]
==split [1] w.r.t ([2], 'a')
new_split_of [1] is [-1]
==split [2] w.r.t ([2], 'a')
new_split_of [2] is [3]
[p] = {2}
[r] = {3}
p and r are the new representatives. Now update L with the smallest of
[2] and [3]
using [p] = [2], L[r] = L[p]; L[p] = C.size();
after update L:
L:
0 1 2 3 4 5 6 7 8
0 0 2 0 0 0 0 0 2
==split [4] w.r.t ([2], 'a')
new_split_of [4] is [-1]
==split [6] w.r.t ([2], 'a')
new_split_of [6] is [-1]

```

```

====split [8] w.r.t ([2], 'a')
new_split_of [8] is [-1]
=====Iterate: k=9
L:
0 1 2 3 4 5 6 7 8
0 0 0 1 0 0 0 0 0 2
Partitions:
StateEqRel
{ 0 }
{ 1 }
{ 2 }
{ 3 }
{ 4 5 }
{ 6 7 }
{ 8 }

pick [q] in L: ([q], a) = ([2], 'b')
split [p] w.r.t ([2], 'b')
====split [0] w.r.t ([2], 'b')
new_split_of [0] is [-1]
====split [1] w.r.t ([2], 'b')
new_split_of [1] is [-1]
====split [2] w.r.t ([2], 'b')
new_split_of [2] is [-1]
====split [3] w.r.t ([2], 'b')
new_split_of [3] is [-1]
====split [4] w.r.t ([2], 'b')
new_split_of [4] is [-1]
====split [6] w.r.t ([2], 'b')
new_split_of [6] is [-1]
====split [8] w.r.t ([2], 'b')
new_split_of [8] is [-1]
=====Iterate: k=10
L:
0 1 2 3 4 5 6 7 8
0 0 0 0 0 0 0 0 0 2
Partitions:
StateEqRel
{ 0 }
{ 1 }
{ 2 }

```

$\{ \sqcup 3 \sqcup \}$

$\{ \sqcup 4 \sqcup \sqcup 5 \sqcup \}$

$\{ \sqcup 6 \sqcup \sqcup 7 \sqcup \}$

$\{ \sqcup 8 \sqcup \}$

$\text{pick}_{\sqcup} [q]_{\sqcup} \text{in}_{\sqcup} L : ([q], a) = ([2], 'a')$

$\text{split}_{\sqcup} [p]_{\sqcup} \text{w.r.t}_{\sqcup} ([2], 'a')$

$\equiv \text{split} [0]_{\sqcup} \text{w.r.t}_{\sqcup} ([2], 'a')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [0]_{\sqcup} \text{is}_{\sqcup} [-1]$

$\equiv \text{split} [1]_{\sqcup} \text{w.r.t}_{\sqcup} ([2], 'a')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [1]_{\sqcup} \text{is}_{\sqcup} [-1]$

$\equiv \text{split} [2]_{\sqcup} \text{w.r.t}_{\sqcup} ([2], 'a')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [2]_{\sqcup} \text{is}_{\sqcup} [-1]$

$\equiv \text{split} [3]_{\sqcup} \text{w.r.t}_{\sqcup} ([2], 'a')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [3]_{\sqcup} \text{is}_{\sqcup} [-1]$

$\equiv \text{split} [4]_{\sqcup} \text{w.r.t}_{\sqcup} ([2], 'a')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [4]_{\sqcup} \text{is}_{\sqcup} [-1]$

$\equiv \text{split} [6]_{\sqcup} \text{w.r.t}_{\sqcup} ([2], 'a')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [6]_{\sqcup} \text{is}_{\sqcup} [-1]$

$\equiv \text{split} [8]_{\sqcup} \text{w.r.t}_{\sqcup} ([2], 'a')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [8]_{\sqcup} \text{is}_{\sqcup} [-1]$

$\equiv \text{Iterate} :_{\sqcup} k_{\sqcup} =_{\sqcup} 11$

L:

$0_{\sqcup} 1_{\sqcup} 2_{\sqcup} 3_{\sqcup} 4_{\sqcup} 5_{\sqcup} 6_{\sqcup} 7_{\sqcup} 8$

$0_{\sqcup} 0_{\sqcup} 0_{\sqcup} 0_{\sqcup} 0_{\sqcup} 0_{\sqcup} 0_{\sqcup} 0_{\sqcup} 1$

Partitions:

StateEqRel

$\{ \sqcup 0 \sqcup \}$

$\{ \sqcup 1 \sqcup \}$

$\{ \sqcup 2 \sqcup \}$

$\{ \sqcup 3 \sqcup \}$

$\{ \sqcup 4 \sqcup \sqcup 5 \sqcup \}$

$\{ \sqcup 6 \sqcup \sqcup 7 \sqcup \}$

$\{ \sqcup 8 \sqcup \}$

$\text{pick}_{\sqcup} [q]_{\sqcup} \text{in}_{\sqcup} L : ([q], a) = ([8], 'b')$

$\text{split}_{\sqcup} [p]_{\sqcup} \text{w.r.t}_{\sqcup} ([8], 'b')$

$\equiv \text{split} [0]_{\sqcup} \text{w.r.t}_{\sqcup} ([8], 'b')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [0]_{\sqcup} \text{is}_{\sqcup} [-1]$

$\equiv \text{split} [1]_{\sqcup} \text{w.r.t}_{\sqcup} ([8], 'b')$

$\text{new}_{\sqcup} \text{split}_{\sqcup} \text{of}_{\sqcup} [1]_{\sqcup} \text{is}_{\sqcup} [-1]$

```

==split [2] w.r.t ([8] , 'b')
new_split_of [2] is [-1]
==split [3] w.r.t ([8] , 'b')
new_split_of [3] is [-1]
==split [4] w.r.t ([8] , 'b')
new_split_of [4] is [-1]
==split [6] w.r.t ([8] , 'b')
new_split_of [6] is [-1]
==split [8] w.r.t ([8] , 'b')
new_split_of [8] is [-1]

=====Iterate : k=12

L:
0 1 2 3 4 5 6 7 8
0 0 0 0 0 0 0 0 0
Partitions :
StateEqRel
{ 0 }
{ 1 }
{ 2 }
{ 3 }
{ 4 5 }
{ 6 7 }
{ 8 }

pick [q] in L : ([q] , a) = ([8] , 'a')
split [p] w.r.t ([8] , 'a')
==split [0] w.r.t ([8] , 'a')
new_split_of [0] is [-1]
==split [1] w.r.t ([8] , 'a')
new_split_of [1] is [-1]
==split [2] w.r.t ([8] , 'a')
new_split_of [2] is [-1]
==split [3] w.r.t ([8] , 'a')
new_split_of [3] is [-1]
==split [4] w.r.t ([8] , 'a')
new_split_of [4] is [-1]
==split [6] w.r.t ([8] , 'a')
new_split_of [6] is [-1]
==split [8] w.r.t ([8] , 'a')
new_split_of [8] is [-1]

```

*****_minDFA

DFA

$Q = [0, 7)$

$S = \{0\}$

$F = \{5, 6\}$

Transitions =

$0 \rightarrow \{ 'a' \rightarrow 1, 'b' \rightarrow 6 \}$

$1 \rightarrow \{ 'a' \rightarrow 2, 'b' \rightarrow 4 \}$

$2 \rightarrow \{ 'a' \rightarrow 2, 'b' \rightarrow 5 \}$

$3 \rightarrow \{ 'a' \rightarrow 4, 'b' \rightarrow 5 \}$

$4 \rightarrow \{ 'a', 'b' \rightarrow 5 \}$

$5 \rightarrow \{ 'a', 'b' \rightarrow 5 \}$

$6 \rightarrow \{ 'a' \rightarrow 4, 'b' \rightarrow 3 \}$

current = 1

1.1 Minimization example 2

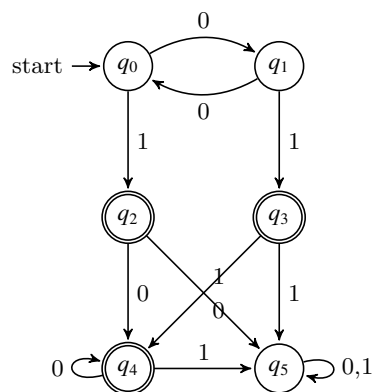


图 1.2: Minimization example 2

DFA

$Q = [0, 6)$

$S = \{0\}$

$F = \{2, 3, 4\}$

Transitions =

$0 \rightarrow \{ '0' \rightarrow 1, '1' \rightarrow 2 \}$

$1 \rightarrow \{ '0' \rightarrow 0, '1' \rightarrow 3 \}$

```

2->{ '0'->4 '1'->5 }
3->{ '0'->4 '1'->5 }
4->{ '0'->4 '1'->5 }
5->{ ['0','1']->5 }

```

```
current = -1
```

```
is the DFA Useful?: 0
```

The combination for all the out labels of State's: $C = \{ '0', '1' \}$

```

L:
0 1 2 3 4 5
0 0 0 0 0 0
Initialize partitions, E0:
StateEqRel
{ 0 1 5 }
{ 2 3 4 }

Initialize Lrepr = {F}:
{ 2 }

L:
0 1 2 3 4 5
0 0 2 0 0 0
===== Iterate: k=1
L:
0 1 2 3 4 5
0 0 1 0 0 0
Partitions:
StateEqRel
{ 0 1 5 }
{ 2 3 4 }

pick [q] in L: ([q], a) = ([2], '1')
split [p] w.r.t ([2], '1')
== split [0] w.r.t ([2], '1')
new split of [0] is [5]
[p] = { 0 1 }
[r] = { 5 }
p and r are the new representatives. Now update L with the
smallest of [0] and [5]

```



```

using [r] = [5], L[r] = C.size();
after update L:
L:
0 1 2 3 4 5
0 0 1 0 0 2
==split [2] w.r.t ([2], '1')
new split of [2] is [-1]
===== Iterate: k = 2
L:
0 1 2 3 4 5
0 0 0 0 0 2
Partitions:
StateEqRel
{ 0 1 }
{ 2 3 4 }
{ 5 }
=====
pick [q] in L: ([q], a) = ([2], '0')
split [p] w.r.t ([2], '0')
==split [0] w.r.t ([2], '0')
new split of [0] is [-1]
==split [2] w.r.t ([2], '0')
new split of [2] is [-1]
==split [5] w.r.t ([2], '0')
new split of [5] is [-1]
===== Iterate: k = 3
L:
0 1 2 3 4 5
0 0 0 0 0 1
Partitions:
StateEqRel
{ 0 1 }
{ 2 3 4 }
{ 5 }
=====
pick [q] in L: ([q], a) = ([5], '1')
split [p] w.r.t ([5], '1')
==split [0] w.r.t ([5], '1')
new split of [0] is [-1]
==split [2] w.r.t ([5], '1')
new split of [2] is [-1]

```

```

=====split [5] w.r.t ([5], '1')
new_split_of [5] is [-1]
=====Iterate: k=4
L:
0 1 2 3 4 5
0 0 0 0 0 0
Partitions:
StateEqRel
{0 1}
{2 3 4}
{5}

pick [q] in L: ([q], a) = ([5], '0')
split [p] w.r.t ([5], '0')
=====split [0] w.r.t ([5], '0')
new_split_of [0] is [-1]
=====split [2] w.r.t ([5], '0')
new_split_of [2] is [-1]
=====split [5] w.r.t ([5], '0')
new_split_of [5] is [-1]

*****minDFA

DFA
Q=[0,3)
S={0}
F={1}
Transitions=
0->{'0'>0, '1'>1}
1->{'0'>1, '1'>2}
2->{'0', '1'>2}

current=-1
\end{list}

```

1.2 Minimization example 3 ($1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$)

split Q_0 w.r.t. $(Q_1, 0)$, see Fig. 1.4.

Ex 1: $Q_0 = \{0, 1, 2, 3, 4\}, Q_1 = \{5\}$

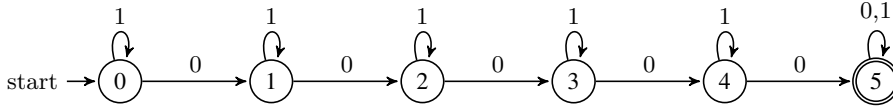


图 1.3: Minimizing example

Since, $Q'_0 = \{4\} \subseteq Q_0$, and $T(Q'_0, 0) = T(\{4\}, 0) = 5 \in Q_1$

$Q''_0 = Q_0 \setminus Q'_0 = \{0, 1, 2, 3\} \subseteq Q_0$, and $T(Q''_0, 0) = T(\{0, 1, 2, 3\}, 0) = T(\{0\}, 0) \cup T(\{1\}, 0) \cup T(\{2\}, 0) \cup T(\{3\}, 0) = \{1\} \cup \{2\} \cup \{3\} = \{1, 2, 3\} \notin Q_1$

$\therefore Q_0$ w.r.t. $(Q_1, 0)$ is splitted into two parts. part(1) $Q'_0 = \{4\}$, part(2) $Q''_0 = Q_0 \setminus Q'_0 = \{0, 1, 2, 3\}$

Ex 2: $Q_0 = \{5\}, Q_1 = \{0, 1, 2, 3, 4\}$

Since, $\forall q \in Q_0, \nexists T(q, a) \in Q_1$

$\therefore Q_0$ w.r.t. $(Q_1, 0)$ 是一个无效的 split, 同样 $Q_0 = Q_1 = \{5\}$ 也是一个无效的 split。

Ex 3: $Q_0 = \{0, 1, 2, 3, 4\}, Q_1 = \{0, 1, 2, 3, 4\}$

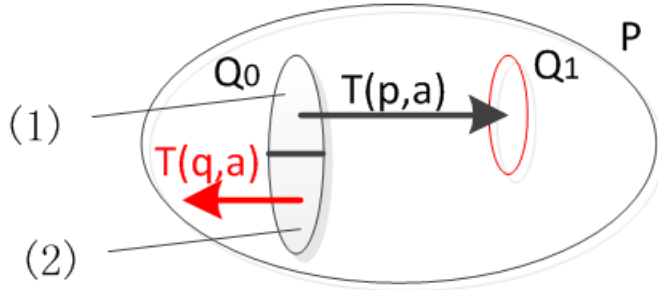
Since, $Q'_0 = \{0, 1, 2, 3\} \subseteq Q_0$, and $T(Q'_0, 0) = T(\{0, 1, 2, 3\}, 0) = T(\{0\}, 0) \cup T(\{1\}, 0) \cup T(\{2\}, 0) \cup T(\{3\}, 0) = \{1\} \cup \{2\} \cup \{3\} = \{1, 2, 3\} \in Q_1$

$Q''_0 = Q_0 \setminus Q'_0 = \{4\} \subseteq Q_0$, and $T(Q''_0, 0) = T(\{4\}, 0) = 5 \notin Q_1$

$\therefore Q_0$ w.r.t. $(Q_1, 0)$ is splitted into two parts. part(1) $Q'_0 = \{0, 1, 2, 3\}$, part(2) $Q''_0 = Q_0 \setminus Q'_0 = \{4\}$

if $(\exists p, q \in Q_0, T(p, a) \in Q_1, \text{ and } T(q, a) \notin Q_1)$, then

split Q_0 w.r.t. $(Q_1, a) \rightarrow$ two parts, (1): $Q'_0 \in Q_1$ and (2): $(Q_0 \setminus Q'_0) \notin Q_1$

图 1.4: split Q_0 w.r.t. (Q_1, a)

开始于: $[q] = \{Q \setminus F\} = \{0, 1, 2, 3, 4\}$, split $[p]$ w.r.t. $([q], a)$

***** DFA

DFA

$Q = [0, 6)$

$S = \{ 0 \}$

```
F = { 5 }
```

```
Transitions =
```

```
0->{ '0'->1 '1'->0 }
```

```
1->{ '0'->2 '1'->1 }
```

```
2->{ '0'->3 '1'->2 }
```

```
3->{ '0'->4 '1'->3 }
```

```
4->{ '0'->5 '1'->4 }
```

```
5->{ [ '0', '1' ]->5 }
```

```
current = -1
```

```
is the DFA Useful?: 1
```

The combination **for** all the out labels of State's: $C = \{ '0', '1' \}$

```
L:
```

```
0 1 2 3 4 5
```

```
0 0 0 0 0 0
```

```
Initialize partitions, E0:
```

```
StateEqRel
```

```
{ 0 1 2 3 4 }
```

```
{ 5 }
```

```
L:
```

```
0 1 2 3 4 5
```

```
2 0 0 0 0 0
```

```
===== Iterate: k=1
```

```
L:
```

```
0 1 2 3 4 5
```

```
1 0 0 0 0 0
```

```
Partitions:
```

```
StateEqRel
```

```
{ 0 1 2 3 4 }
```

```
{ 5 }
```

```
pick [q] in L: ([q], a) = ([0], '1')
```

```
split [p] w.r.t ([0], '1')
```

```
== split [0] w.r.t ([0], '1')
```

```
new split of [0] is [-1]
```

```
== split [5] w.r.t ([0], '1')
```

```
new split of [5] is [-1]
```

```
===== Iterate: k=2
```

L:

0 1 2 3 4 5

0 0 0 0 0 0

Partitions:

StateEqRel

{ 0 1 2 3 4 }

{ 5 }

pick [q] in L: ([q], a) = ([0], '0')

split [p] w.r.t ([0], '0')

== split [0] w.r.t ([0], '0')

new split of [0] is [4]

[p] = { 0 1 2 3 }

[r] = { 4 }

p and r are the new representatives. Now update L with the smallest of [0] and [4]

using [r] = [4], L[r] = C.size();

after update L:

L:

0 1 2 3 4 5

0 0 0 0 2 0

== split [5] w.r.t ([0], '0')

new split of [5] is [-1]

===== Iterate: k = 3

L:

0 1 2 3 4 5

0 0 0 0 1 0

Partitions:

StateEqRel

{ 0 1 2 3 }

{ 4 }

{ 5 }

pick [q] in L: ([q], a) = ([4], '1')

split [p] w.r.t ([4], '1')

== split [0] w.r.t ([4], '1')

new split of [0] is [-1]

== split [4] w.r.t ([4], '1')

new split of [4] is [-1]

== split [5] w.r.t ([4], '1')

new split of [5] is [-1]

```

=====Iterate: k=4

L:
0 1 2 3 4 5
0 0 0 0 0 0
Partitions:
StateEqRel
{ 0 1 2 3 }
{ 4 }
{ 5 }

pick [q] in L: ([q], a) = ([4], '0')
split [p] w.r.t ([4], '0')
==split [0] w.r.t ([4], '0')
new split of [0] is [3]
[p] = { 0 1 2 }
[r] = { 3 }
p and r are the new representatives. Now update L with the smallest of
    [0] and [3]
using [r] = [3], L[r] = C.size();
after update L:
L:
0 1 2 3 4 5
0 0 0 2 0 0
==split [4] w.r.t ([4], '0')
new split of [4] is [-1]
==split [5] w.r.t ([4], '0')
new split of [5] is [-1]
=====Iterate: k=5

L:
0 1 2 3 4 5
0 0 0 1 0 0
Partitions:
StateEqRel
{ 0 1 2 }
{ 3 }
{ 4 }
{ 5 }

pick [q] in L: ([q], a) = ([3], '1')
split [p] w.r.t ([3], '1')
==split [0] w.r.t ([3], '1')

```

```

new_split_of [0] is [-1]
==split [3] w.r.t ([3], '1')
new_split_of [3] is [-1]
==split [4] w.r.t ([3], '1')
new_split_of [4] is [-1]
==split [5] w.r.t ([3], '1')
new_split_of [5] is [-1]
=====Iterate: k=6

L:
0 1 2 3 4 5
0 0 0 0 0 0
Partitions:
StateEqRel
{0 1 2}
{3}
{4}
{5}

pick [q] in L: ([q], a) = ([3], '0')
split [p] w.r.t ([3], '0')
==split [0] w.r.t ([3], '0')
new_split_of [0] is [2]
[p] = {0 1}
[r] = {2}
p and r are the new representatives. Now update L with the smallest of
[0] and [2]
using [r] = [2], L[r] = C.size();
after update L:
L:
0 1 2 3 4 5
0 0 2 0 0 0
==split [3] w.r.t ([3], '0')
new_split_of [3] is [-1]
==split [4] w.r.t ([3], '0')
new_split_of [4] is [-1]
==split [5] w.r.t ([3], '0')
new_split_of [5] is [-1]
=====Iterate: k=7

L:
0 1 2 3 4 5
0 0 1 0 0 0

```



```

after_update_L:
L:
0_1_2_3_4_5
2_0_0_0_0_0
==split [2]_w.r.t_([2], '0')
new_split_of [2]_is [-1]
==split [3]_w.r.t_([2], '0')
new_split_of [3]_is [-1]
==split [4]_w.r.t_([2], '0')
new_split_of [4]_is [-1]
==split [5]_w.r.t_([2], '0')
new_split_of [5]_is [-1]
=====Iterate: k=9

```

```

L:
0_1_2_3_4_5
1_0_0_0_0_0
Partitions:
StateEqRel
{0_}
{1_}
{2_}
{3_}
{4_}
{5_}

```

```

pick [q]_in_L: ([q], a) = ([0], '1')
split [p]_w.r.t_([0], '1')
==split [0]_w.r.t_([0], '1')
new_split_of [0]_is [-1]
==split [1]_w.r.t_([0], '1')
new_split_of [1]_is [-1]
==split [2]_w.r.t_([0], '1')
new_split_of [2]_is [-1]
==split [3]_w.r.t_([0], '1')
new_split_of [3]_is [-1]
==split [4]_w.r.t_([0], '1')
new_split_of [4]_is [-1]
==split [5]_w.r.t_([0], '1')
new_split_of [5]_is [-1]
=====Iterate: k=10

```

```

L:

```

```
0_1_2_3_4_5
```

```
0_0_0_0_0_0
```

```
Partitions:
```

```
StateEqRel
```

```
{_0_}
```

```
{_1_}
```

```
{_2_}
```

```
{_3_}
```

```
{_4_}
```

```
{_5_}
```

```
pick_[q]_in_L:([q],a)=([0], '0')
```

```
split_[p]_w.r.t_([0], '0')
```

```
==split [0]_w.r.t_([0], '0')
```

```
new_split_of_[0]_is_-1
```

```
==split [1]_w.r.t_([0], '0')
```

```
new_split_of_[1]_is_-1
```

```
==split [2]_w.r.t_([0], '0')
```

```
new_split_of_[2]_is_-1
```

```
==split [3]_w.r.t_([0], '0')
```

```
new_split_of_[3]_is_-1
```

```
==split [4]_w.r.t_([0], '0')
```

```
new_split_of_[4]_is_-1
```

```
==split [5]_w.r.t_([0], '0')
```

```
new_split_of_[5]_is_-1
```

```
*****_minDFA
```

```
DFA
```

```
Q_=[0,6)
```

```
S_={_0_}
```

```
F_={_5_}
```

```
Transitions_
```

```
0->{_0'->1_ '1'->0_}
```

```
1->{_0'->2_ '1'->1_}
```

```
2->{_0'->3_ '1'->2_}
```

```
3->{_0'->4_ '1'->3_}
```

```
4->{_0'->5_ '1'->4_}
```

```
5->{_'0','1']->5_}
```

```
current_=-1
```

开始于: $[q] = \{F\} = \{5\}$, split $[p]$ w.r.t. $([q], a)$

***** DFA

DFA

$Q = [0, 6)$

$S = \{ 0 \}$

$F = \{ 5 \}$

Transitions =

$0 \rightarrow \{ '0' \rightarrow 1 \quad '1' \rightarrow 0 \}$

$1 \rightarrow \{ '0' \rightarrow 2 \quad '1' \rightarrow 1 \}$

$2 \rightarrow \{ '0' \rightarrow 3 \quad '1' \rightarrow 2 \}$

$3 \rightarrow \{ '0' \rightarrow 4 \quad '1' \rightarrow 3 \}$

$4 \rightarrow \{ '0' \rightarrow 5 \quad '1' \rightarrow 4 \}$

$5 \rightarrow \{ ['0', '1'] \rightarrow 5 \}$

current = -1

is the DFA Useful?: 1

The combination for all the out labels of State's: $C = \{ '0', '1' \}$

L:

0 1 2 3 4 5

0 0 0 0 0 0

Initialize partitions, E0:

StateEqRel

$\{ 0 \quad 1 \quad 2 \quad 3 \quad 4 \}$

$\{ 5 \}$

Initialize Lrepr = {F}:

$\{ 5 \}$

L:

0 1 2 3 4 5

0 0 0 0 0 2

===== Iterate: k = 1

L:

0 1 2 3 4 5

0 0 0 0 0 1

Partitions:

StateEqRel

$\{ 0 \quad 1 \quad 2 \quad 3 \quad 4 \}$

$\{ _5 \}$

$\text{pick_}[q] _ \text{in_} L : ([q], a) = ([5], '1')$

$\text{split_}[p] _ \text{w.r.t_} ([5], '1')$

$\equiv \text{split } [0] _ \text{w.r.t_} ([5], '1')$

$\text{new_split_of_} [0] _ \text{is_} [-1]$

$\equiv \text{split } [5] _ \text{w.r.t_} ([5], '1')$

$\text{new_split_of_} [5] _ \text{is_} [-1]$

$\equiv \text{Iterate : } _ k = 2$

L:

$0_1_2_3_4_5$

$0_0_0_0_0_0$

Partitions:

StateEqRel

$\{ _0_1_2_3_4 \}$

$\{ _5 \}$

$\text{pick_}[q] _ \text{in_} L : ([q], a) = ([5], '0')$

$\text{split_}[p] _ \text{w.r.t_} ([5], '0')$

$\equiv \text{split } [0] _ \text{w.r.t_} ([5], '0')$

$\text{new_split_of_} [0] _ \text{is_} [4]$

$[p] = \{ _0_1_2_3 \}$

$[r] = \{ _4 \}$

p and r are the new representatives. Now update L with the smallest of $[0]$ and $[4]$

using $[r] = [4]$, $L[r] = C.size()$;

after update L :

L:

$0_1_2_3_4_5$

$0_0_0_0_2_0$

$\equiv \text{split } [5] _ \text{w.r.t_} ([5], '0')$

$\text{new_split_of_} [5] _ \text{is_} [-1]$

$\equiv \text{Iterate : } _ k = 3$

L:

$0_1_2_3_4_5$

$0_0_0_0_1_0$

Partitions:

StateEqRel

$\{ _0_1_2_3 \}$

$\{ _4 \}$

$\{ _5 \}$

```

pick [q] in L: ([q], a) = ([4], '1')
split [p] w.r.t ([4], '1')
==split [0] w.r.t ([4], '1')
new split of [0] is [-1]
==split [4] w.r.t ([4], '1')
new split of [4] is [-1]
==split [5] w.r.t ([4], '1')
new split of [5] is [-1]
===== Iterate: k=4

L:
0 1 2 3 4 5
0 0 0 0 0 0
Partitions:
StateEqRel
{ 0 1 2 3 }
{ 4 }
{ 5 }

pick [q] in L: ([q], a) = ([4], '0')
split [p] w.r.t ([4], '0')
==split [0] w.r.t ([4], '0')
new split of [0] is [3]
[p] = { 0 1 2 }
[r] = { 3 }
p and r are the new representatives. Now update L with the smallest of
[0] and [3]
using [r] = [3], L[r] = C.size();
after update L:
L:
0 1 2 3 4 5
0 0 0 2 0 0
==split [4] w.r.t ([4], '0')
new split of [4] is [-1]
==split [5] w.r.t ([4], '0')
new split of [5] is [-1]
===== Iterate: k=5

L:
0 1 2 3 4 5
0 0 0 1 0 0
Partitions:

```

StateEqRel

{012}

{3}

{4}

{5}

pick [q] in L: ([q], a) = ([3], '1')

split [p] w.r.t ([3], '1')

== split [0] w.r.t ([3], '1')

new split of [0] is [-1]

== split [3] w.r.t ([3], '1')

new split of [3] is [-1]

== split [4] w.r.t ([3], '1')

new split of [4] is [-1]

== split [5] w.r.t ([3], '1')

new split of [5] is [-1]

==== Iterate: k=6

L:

012345

000000

Partitions:

StateEqRel

{012}

{3}

{4}

{5}

pick [q] in L: ([q], a) = ([3], '0')

split [p] w.r.t ([3], '0')

== split [0] w.r.t ([3], '0')

new split of [0] is [2]

[p] = {01}

[r] = {2}

p and r are the new representatives. Now update L with the smallest of [0] and [2]

using [r] = [2], L[r] = C.size();

after update L:

L:

012345

002000

== split [3] w.r.t ([3], '0')

```

new_split_of [3] is [-1]
==split [4] w.r.t ([3], '0')
new_split_of [4] is [-1]
==split [5] w.r.t ([3], '0')
new_split_of [5] is [-1]
=====Iterate: k=7

```

L:

0 1 2 3 4 5

0 0 1 0 0 0

Partitions:

StateEqRel

{ 0 1 }

{ 2 }

{ 3 }

{ 4 }

{ 5 }

pick [q] in L: ([q], a) = ([2], '1')

split [p] w.r.t ([2], '1')

==split [0] w.r.t ([2], '1')

new_split_of [0] is [-1]

==split [2] w.r.t ([2], '1')

new_split_of [2] is [-1]

==split [3] w.r.t ([2], '1')

new_split_of [3] is [-1]

==split [4] w.r.t ([2], '1')

new_split_of [4] is [-1]

==split [5] w.r.t ([2], '1')

new_split_of [5] is [-1]

=====Iterate: k=8

L:

0 1 2 3 4 5

0 0 0 0 0 0

Partitions:

StateEqRel

{ 0 1 }

{ 2 }

{ 3 }

{ 4 }

{ 5 }

```

pick [q] in L: ([q], a) = ([2], '0')
split [p] w.r.t ([2], '0')
==split [0] w.r.t ([2], '0')
new split of [0] is [1]
[p] = { [0] }
[r] = { [1] }
p and r are the new representatives. Now update L with the smallest of
[0] and [1]
using [p] = [0], L[r] = L[p]; L[p] = C.size();
after update L:
L:
0 1 2 3 4 5
2 0 0 0 0 0
==split [2] w.r.t ([2], '0')
new split of [2] is [-1]
==split [3] w.r.t ([2], '0')
new split of [3] is [-1]
==split [4] w.r.t ([2], '0')
new split of [4] is [-1]
==split [5] w.r.t ([2], '0')
new split of [5] is [-1]
===== Iterate: k = 9
L:
0 1 2 3 4 5
1 0 0 0 0 0
Partitions:
StateEqRel
{ [0] }
{ [1] }
{ [2] }
{ [3] }
{ [4] }
{ [5] }

pick [q] in L: ([q], a) = ([0], '1')
split [p] w.r.t ([0], '1')
==split [0] w.r.t ([0], '1')
new split of [0] is [-1]
==split [1] w.r.t ([0], '1')
new split of [1] is [-1]
==split [2] w.r.t ([0], '1')

```



```

new_split_of [2] is [-1]
==split [3] w.r.t ([0], '1')
new_split_of [3] is [-1]
==split [4] w.r.t ([0], '1')
new_split_of [4] is [-1]
==split [5] w.r.t ([0], '1')
new_split_of [5] is [-1]
=====Iterate:k=10

```

L:

0 1 2 3 4 5

0 0 0 0 0 0

Partitions:

StateEqRel

{0}

{1}

{2}

{3}

{4}

{5}

pick[q] in L: ([q], a) = ([0], '0')

split[p] w.r.t ([0], '0')

==split [0] w.r.t ([0], '0')

new_split_of [0] is [-1]

==split [1] w.r.t ([0], '0')

new_split_of [1] is [-1]

==split [2] w.r.t ([0], '0')

new_split_of [2] is [-1]

==split [3] w.r.t ([0], '0')

new_split_of [3] is [-1]

==split [4] w.r.t ([0], '0')

new_split_of [4] is [-1]

==split [5] w.r.t ([0], '0')

new_split_of [5] is [-1]

*****minDFA

DFA

Q=[0,6)

S={0}

F={5}

```

Transitions□=
0->{□'0'->1□□'1'->0□}
1->{□'0'->2□□'1'->1□}
2->{□'0'->3□□'1'->2□}
3->{□'0'->4□□'1'->3□}
4->{□'0'->5□□'1'->4□}
5->{□['0','1']->5□}

current□=□-1

```

开始于: $[q] = \{Q \setminus F\} = \{0, 1, 2, 3, 4\}$, split $[p]$ w.r.t. $([q], a)$
或者开始于 $[q] = \{F\} = \{5\}$, split $[p]$ w.r.t. $([q], a)$ 处理结果是一致的。

1.3 Minimization example 4

$\{0, 1\}, \{3, 4\}$ is not equivalent states.

Sets of equivalent states: $\{0, 2\}, \{1\}, \{3\}, \{4\}$

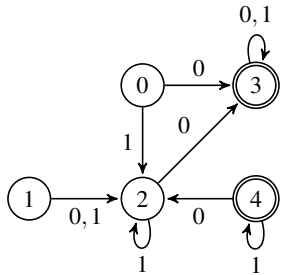


图 1.5: Finite state automaton

Partitions: $\{ \{0, 1, 2\} \{3, 4\} \}$ $Q_0 = Q_1 = \{3, 4\}$ split Q_0 w.r.t. $(Q_1, '1')$ is invalid.
Since $T(\{3, 4\}, '1') = \{3, 4\} = Q' = Q_1$, but $Q'' = Q_0 \setminus Q'_0 = \emptyset$
so, split $[3]$ w.r.t. $([3], '1')$ is invalid.

***** DFA

DFA

$Q = [0, 5)$

$S = \{ 0 \}$

$F = \{ 3, 4 \}$

Transitions =

$0 \rightarrow \{ '0' \rightarrow 3 \quad '1' \rightarrow 2 \}$

```

1->{ ['0','1']->2 }
2->{ '0'->3  '1'->2 }
3->{ ['0','1']->3 }
4->{ '0'->2  '1'->4 }

```

```
current = -1
```

```
is the DFA Useful?: 1
```

The combination `for` all the out labels of State'súC₀={₀'0'₀₀'1'₀}

```

00000000L:
000000000_1_2_3_4
000000000_0_0_0_0_0
00000000Initialize_partitions,_E0:
00000000StateEqRel
00000000{_0_1_2_}
00000000{_3_4_}

00000000
00000000Initialize_L_repr={F}:
00000000{_3_}

00000000L:
000000000_1_2_3_4
000000000_0_0_0_2_0

00000000=====Iterate:_k_=1

00000000L:
000000000_1_2_3_4
000000000_0_0_0_1_0
00000000Partitions:
00000000StateEqRel
00000000{_0_1_2_}
00000000{_3_4_}

00000000
00000000pick_[q]_in_L:([q],a)=([3], '1')
00000000split_[p]_w.r.t_([3], '1')
00000000==split[0]_w.r.t_([3], '1')
00000000new_split_of_[0]_is_[-1]
00000000==split[3]_w.r.t_([3], '1')
00000000new_split_of_[3]_is_[-1]

00000000=====Iterate:_k_=2

00000000L:

```

```

0 1 2 3 4
0 0 0 0 0
Partitions:
StateEqRel
{ 0 1 2 }
{ 3 4 }

pick [q] in L: ([q], a) = ([3], '0')
split [p] w.r.t ([3], '0')
==split [0] w.r.t ([3], '0')
new split of [0] is [1]
[p] = { 0 2 }
[r] = { 1 }
p and r are the new representatives. Now update L with the
smallest of [0] and [1]
using [r] = [1], L[r] = C.size();
after update L:
L:
0 1 2 3 4
0 2 0 0 0
==split [3] w.r.t ([3], '0')
new split of [3] is [4]
[p] = { 3 }
[r] = { 4 }
p and r are the new representatives. Now update L with the
smallest of [3] and [4]
using [p] = [3], L[r] = L[p]; L[p] = C.size();
after update L:
L:
0 1 2 3 4
0 2 0 2 0
===== Iterate: k = 3
L:
0 1 2 3 4
0 1 0 2 0
Partitions:
StateEqRel
{ 0 2 }
{ 1 }
{ 3 }
{ 4 }

```

```

#####
#####pick [q] in L: ([q], a) = ([1], '1')
#####split [p] w.r.t ([1], '1')
#####==split [0] w.r.t ([1], '1')
#####new split of [0] is [-1]
#####==split [1] w.r.t ([1], '1')
#####new split of [1] is [-1]
#####==split [3] w.r.t ([1], '1')
#####new split of [3] is [-1]
#####==split [4] w.r.t ([1], '1')
#####new split of [4] is [-1]
#####=====Iterate: k=4
#####L:
#####0 1 2 3 4
#####0 0 0 2 0
#####Partitions:
#####StateEqRel
#####{ 0 2 }
#####{ 1 }
#####{ 3 }
#####{ 4 }
#####
#####pick [q] in L: ([q], a) = ([1], '0')
#####split [p] w.r.t ([1], '0')
#####==split [0] w.r.t ([1], '0')
#####new split of [0] is [-1]
#####==split [1] w.r.t ([1], '0')
#####new split of [1] is [-1]
#####==split [3] w.r.t ([1], '0')
#####new split of [3] is [-1]
#####==split [4] w.r.t ([1], '0')
#####new split of [4] is [-1]
#####=====Iterate: k=5
#####L:
#####0 1 2 3 4
#####0 0 0 1 0
#####Partitions:
#####StateEqRel
#####{ 0 2 }
#####{ 1 }
#####{ 3 }

```

```

{}4{}
{}
{}pick [q] in L: ([q], a) = ([3], '1')
{}split [p] w.r.t ([3], '1')
{}==split [0] w.r.t ([3], '1')
{}new split of [0] is [-1]
{}==split [1] w.r.t ([3], '1')
{}new split of [1] is [-1]
{}==split [3] w.r.t ([3], '1')
{}new split of [3] is [-1]
{}==split [4] w.r.t ([3], '1')
{}new split of [4] is [-1]
{}=====Iterate: k=6
{}L:
{}0 1 2 3 4
{}0 0 0 0 0
{}Partitions:
{}StateEqRel
{}{0 2}
{}{1}
{}{3}
{}{4}
{}
{}pick [q] in L: ([q], a) = ([3], '0')
{}split [p] w.r.t ([3], '0')
{}==split [0] w.r.t ([3], '0')
{}new split of [0] is [-1]
{}==split [1] w.r.t ([3], '0')
{}new split of [1] is [-1]
{}==split [3] w.r.t ([3], '0')
{}new split of [3] is [-1]
{}==split [4] w.r.t ([3], '0')
{}new split of [4] is [-1]
{}
{}*****minDFA
{}
{}DFA
{}Q=[0,4)
{}S={0}
{}F={2 3}
{}Transitions=

```

```

0->{ [ ' 0 ' -> 2 [ ' 1 ' -> 0 [
1->{ [ ' 0 ' , ' 1 ' ] -> 0 [
2->{ [ ' 0 ' , ' 1 ' ] -> 2 [
3->{ [ ' 0 ' -> 0 [ ' 1 ' -> 3 [

current [ = [ -1

```

References

- Hopcroft2008. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman 著, 孙家骢等译, 自动机理论、语言和计算机导论, Third Edition, 机械工业出版社, 2008.7
- WATSON93a. WATSON, B. W. *A taxonomy of finite automata construction algorithms*, Computing Science Note 93/43, Eindhoven University of Technology, The Netherlands, 1993. Available by ftp from ftp.win.tue.nl in pub/techreports/pi.
- WATSON93b. WATSON, B. W. *A taxonomy of finite automata minimization algorithms*, Computing Science Note 93/44, Eindhoven University of Technology, The Netherlands, 1993. Available by ftp from ftp.win.tue.nl in pub/techreports/pi.
- WATSON94a. WATSON, B. W. *An introduction to the FIRE engine: A C++ toolkit for FInite automata and Regular Expressions*, Computing Science Note 94/21, Eindhoven University of Technology, The Netherlands, 1994. Available by ftp from ftp.win.tue.nl in pub/techreports/pi
- WATSON94b. WATSON, B.W. *The design. and implementation of the FIRE engine: A C++ toolkit for FInite automata and Regular Expressions*, Computing Science Note 94/22, Eindhoven University of Technology, The Netherlands, 1994. Available by ftp from ftp.win.tue.nl in pub/techreports/pi.
- Chrison2007. Christos G. Cassandras and Stéphane Lafortune, *Introduction to Discrete Event Systems*, Second Edition, New York, Springer, 2007
- Wonham2018. W. M. Wonham and Kai Cai, *Supervisory Control of Discrete-Event Systems*, Revised 2018.01.01
- Jean2018. Jean-Éric Pin, *Mathematical Foundations of Automata Theory*, Version of June 15, 2018
- 蒋宗礼 2013. 蒋宗礼, 姜守旭, 形式语言与自动机理论 (第 3 版), 清华大学出版社, 2013.05
- Lipschutz2007. S. Lipschutz and M. L. Lipson, *Schaum's Outline of Theory and Problems of Discrete Mathematics*, Third Edition, New York: McGraw-Hill, 2007.
- Rosen2007. K. H. Rosen, *Discrete Mathematics and Its Applications*, Seventh Edition, New York: McGraw-Hill, 2007.
- R.Su and Wonham2004. R. Su and W. M. Wonham, *Supervisor reduction for discrete-event systems*, Discrete Event Dyn. Syst., vol. 14, no. 1, pp. 31-53, Jan. 2004.
- Hopcroft71. Hopcroft, J.E. *An $n \log n$ algorithm for minimizing states in a finite automaton*, in The Theory of Machines and Computations (Z. Kohavi, ed.), pp.180-196, Academic Press, New York, 1971.
- Gries73. Gries, D. *Describing an Algorithm by Hopcroft*, Acta Inf. 2:97 109, 173. © by Springer-Verlag 1973
- Knuutila2001. Knuutila, T. *Re-describing an Algorithm by Hopcroft*. Theoret. Computer Science 250 (2001) 333-363.
- Ratnesh95. Ratnesh Kumar, *Modeling and Control of Logical Discrete Event Systems*, © 1995 by Springer Science+Business Media New York.
- Jean2011. Jean Berstel, Luc Boasson, Olivier Carton, Isabelle Fagnot, *Minimization of automata*, Université Paris-Est Marne-la-Vallée 2010 Mathematics Subject Classification: 68Q45, 2011.
- Kenneth2012. Kenneth H. Rosen 著, 徐六通译, 离散数学及其应用 *Discrete Mathematics and Its Applications*, seventh Edition, 2012, 机械工业出版社, 北京, 2014.