目录

Chapter 1
Hopcroft's algorithm

1.0.1 algorithm

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

Algorithm 1 Hopcroft's minimization algorithm

Input: $G = (Q, V, T, q_0, F)$

Output: The equivalence classes of Q

1: $P \leftarrow [Q]_{E_0} = \{F, Q \setminus F\}$ ▷ The initial partitions is $[Q]_{E_0}$, it's the total euivalence relation.
2: $L \leftarrow 0$ ▷ The waiting set
3: **for all** $a \in V$ **do**
4: $ADD((min(F, Q \setminus F), a), L)$ ▷ initialization of the waiting set
5: **end for**
6: **while** $L \neq \emptyset$ **do**
7: $P_{old} = P$;
8: $(Q_1, a) \leftarrow TakeSome(L)$ ▷ 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) ▷ 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: $P = P \setminus \{Q_0\} \cup \{Q'_0, Q''_0\}$ ▷ Refine the partition, Replace Q_0 by Q'_0 and Q''_0 in P.
15: **for all** $b \in V$ **do** ▷ Update the waiting set
16: **if** $(Q_0, b) \in L$ **then**
17: $L = L \setminus \{(Q_0, b)\} \cup \{(Q'_0, b), (Q''_0, b)\}$ ▷ 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**

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):

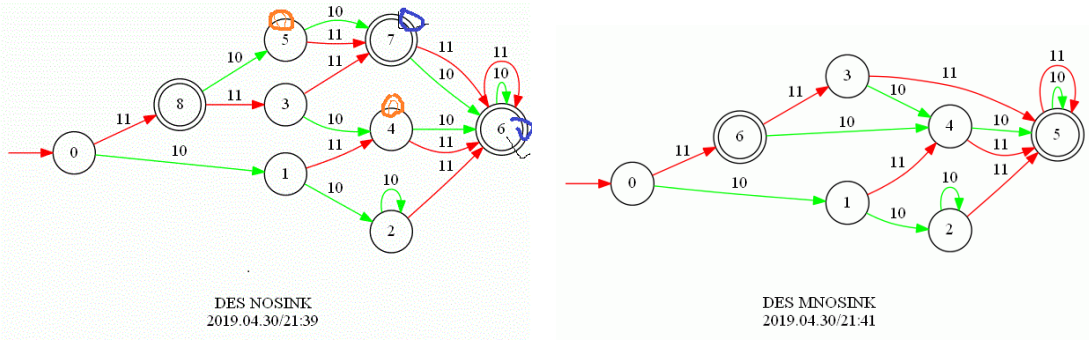
$$([q],C_0),\cdots,([q],C_{L(q)-1})$$

The remaining pairs do not require processing:

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

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

1.0.2 Minimization example 1



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

(b) mini-DFA

图 1.1: 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 }
F = { 6  7  8 }
Transitions =
0->{ 'a'->1  'b'->8 }
1->{ 'a'->2  'b'->4 }
2->{ 'a'->2  'b'->6 }
3->{ 'a'->4  'b'->7 }
4->{ ['a','b']->6 }
```

Algorithm 2 Hopcroft's minimization algorithm

Input: $G = (Q, V, T, q_0, F)$

Output: The equivalence classes of Q

$P \leftarrow [Q]_{E_0} = \{F, Q \setminus F\}$ ▷ The initial partitions is $[Q]_{E_0}$, it's the total euivalence relation.
$L \leftarrow 0$ ▷ The waiting set
$C = V$ ▷ C is all symbols set
if $|F| \leq |Q \setminus F|$ **then** ▷ initialization of the waiting set
    $L[q] = C.Size(), [q]$ is the representive of the F
else
    $L[q] = C.Size(), [q]$ is the representive of the $Q \setminus F$
end if
while (1) **do**
    if all L[q]=0 **then**
        break;
    end if
    Find the first pair in L that still needs processing. $(Q_1, a) = [q], L[q] \neq 0$ ▷ Take and remove some splitter
    $P_{old} = P$ ▷ current partitions
    $L[q]--;$ ▷ Mark this element of L as processed.
    for all $Q_0 \in P_{old}$ **do**
        Q_0 is split by (Q_1, a) ▷ Compute the split, Q_0 is splitted into Q_0' and Q_0''
        $Q_0' = \{p | p \in Q_0 \wedge T(p, a) \in Q_1\}$
        $Q_0'' = \{Q_0 \setminus Q_0'\}$
        $P = P \setminus \{Q_0\} \cup \{Q_0', Q_0''\}$ ▷ Refine the partition, Replace Q_0 by Q_0' and Q_0'' in P.
        $p = Q_0$
        $r = Q_0'$
        if $[r]! = Invalid$ **then** ▷ Update the waiting set
            if $([p] \leq |[r]|)$ **then**
                $L[r] = L[p]$ ▷ [r] 待处理 L[p] 剩下的字符
                $L[p] = C.size()$ ▷ 新的 [p], 待处理 C[0]...C[C.size()-1]
            else
                $L[r] = C.size()$ ▷ // 新的 [r]，待处理 C[0]...C[C.size()-1]
            end if
        end if
    end for
end while

```
5->{ ['a','b']->7 }
6->{ ['a','b']->6 }
7->{ ['a','b']->6 }
8->{ 'a'->5  'b'->3 }

current = -1

is the DFA Usefulf ?: 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 L repr = {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 }

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

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

pick [q] in L:([q],a)=([0],'b')
split [p] w.r.t ([0], 'b')
===split [0] w.r.t ([0], 'b')
new split of [0] is [-1]
===split [1] w.r.t ([0], 'b')
new split of [1] is [-1]
===split [2] 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 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 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 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 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␣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 }
{ 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 [-1]
==split [3] w.r.t ([2], 'a')
new split of [3] is [-1]
==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 = 11
L:
0 1 2 3 4 5 6 7 8
0 0 0 0 0 0 0 0 0 1
Partitions:
StateEqRel
{ 0 }
{ 1 }
{ 2 }
{ 3 }
{ 4  5 }
{ 6  7 }
{ 8 }

pick [q] in L:([q],a)=([8], 'b')
split [p] w.r.t ([8], 'b')
==split [0] w.r.t ([8], 'b')
new split of [0] is [-1]
==split [1] w.r.t ([8], 'b')
new split of [1] is [-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->{ 'a'->1  'b'->6 }
1->{ 'a'->2  'b'->4 }
2->{ 'a'->2  'b'->5 }
3->{ 'a'->4  'b'->5 }
4->{ ['a','b']->5 }
5->{ ['a','b']->5 }
6->{ 'a'->4  'b'->3 }

current = -1
```

1.1 Minimization example 2

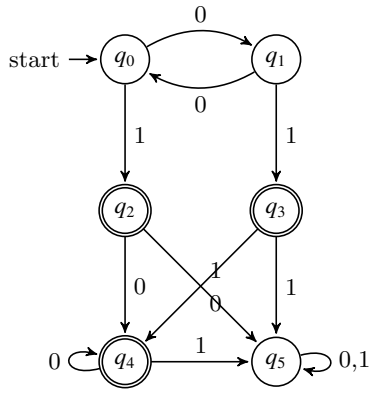


图 1.2: Minimization example 2

```
DFA
Q = [0,6)
S = { 0 }
F = { 2  3  4 }
Transitions =
0->{ '0'->1  '1'->2 }
1->{ '0'->0  '1'->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 Usefulf ?: 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 L repr = {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 ($1 \to 2 \to 3 \to 4 \to 5 \to 6$)

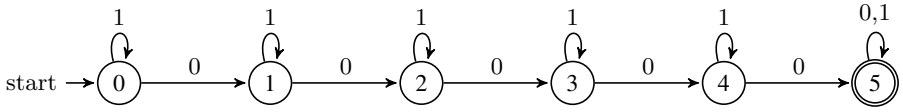


图 1.3: Minimizing example

split Q_0 wrt $(Q_1, 0)$, see Fig. 1.4.
Ex 1: $Q_0 = \{0,1,2,3,4\}, Q_1 = \{5\}$

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 wrt $(Q_1,0)$ 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 wrt $(Q_1,0)$ 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 wrt $(Q_1,0)$ is splitted into two parts. part(1) $Q_0' = \{0,1,2,3\}$, part(2) $Q_0'' = Q_0 \setminus Q_0' = \{4\}$

if $(\exists p,q \in Q_0, T(p,a) \in Q_1$, and $T(q,a) \notin Q_1)$, then
split Q_0 wrt $(Q_1,a) \to$ two parts, (1): $Q_0' \in Q_1$ and (2): $(Q_0 \setminus Q_0') \notin Q_1$

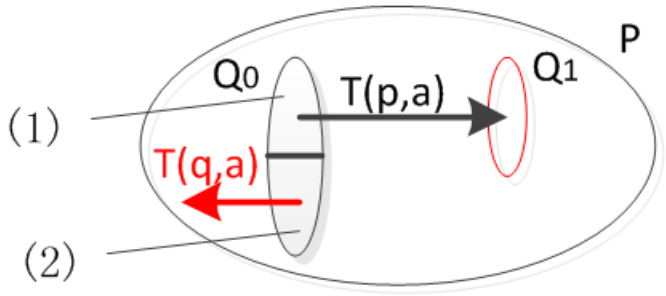


图 1.4: split Q_0 wrt Q_1

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 Usefulf ?: 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
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：$[q] = \{F\} = \{5\}$, 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 Usefulf ?: 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}

Initialize L repr = {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 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 [5] w.r.t ([5], '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)=([5],'0')
split [p] w.r.t ([5],'0')
===split [0] w.r.t ([5],'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 ([5],'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
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

{a,b},{d,e}is not equivalent states.
Sets of equivalent states: {a,c},{b},{d},{e}

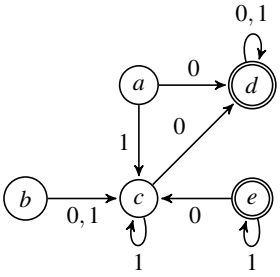


图 1.5: Finite state automaton

References

Hopcroft2008. John E. Hopcroft,Rajeev Motwani,Jeffrey D. Ullman 著, 孙家骕等译, 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*, Universit*é* Paris-Est Marne-la-Vall*é*e 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.