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

$P \leftarrow [Q]_{E_0} = \{F, Q \setminus F\}$ ▷ The initial partitions is $[Q]_{E_0}$, it's the total euivalence relation.
$L \leftarrow 0$ ▷ The waiting set
for all $a \in V$ **do**
    $ADD((min(F, Q \setminus F), a), L)$ ▷ initialization of the waiting set
end for
while $L \neq \emptyset$ **do**
    $P_{old} = P$;
    $(Q_1, a) \leftarrow TakeSome(L)$ ▷ Take and remove some splitter
    $L = L \setminus \{(Q_1, a)\}$;
    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.
        for all $b \in V$ **do** ▷ Update the waiting set
            if $(Q_0, b) \in L$ **then**
                $L = L \setminus \{(Q_0, b)\} \cup \{(Q_0', b), (Q_0'', b)\}$ ▷ Replace (Q_0, b) by (Q_0', b) and (Q_0'', b) in L
            else
                $ADD((min(Q', Q''), b), L)$
            end if
        end for
    end for
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):

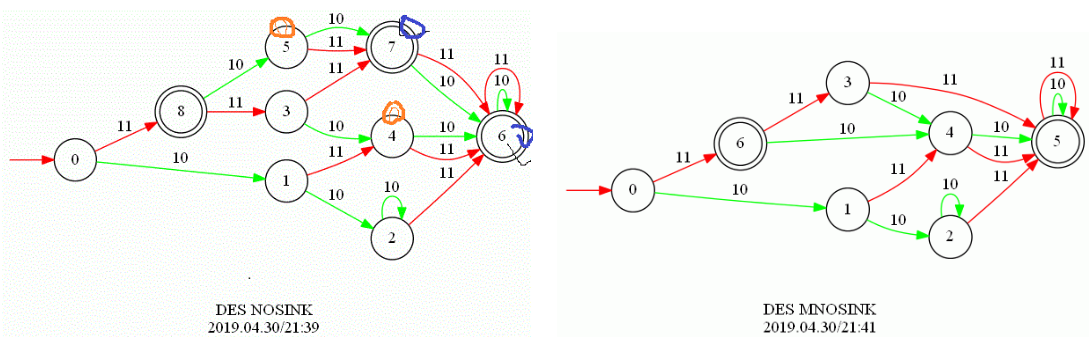
$$([q], C_0), \cdots, ([q], C_{L(q)-1})$$

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 Example



(a) DFA (b) mini-DFA

图 1.1: example minimization

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

int L10]: // 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。 $L = \{0,0,0,0,0,0,0,0,0\}$

Initialize P to be the total equivalence relation $E_0 = \{Q \setminus F, F\}$: $P = \{\{0,1,2,3,4,5\},\{6,7,8\}\}$;

$F.size <= (Q.size() - F.size()$

$L = \{0,0,0,0,0,0,2,0,0\}$ // L[6] represented eq. class in P

— while each [q], (split [p] w.r.t ([q],a))

$[q] = [6]$ // Pick one [q] in L, Processing [6] in current partitions { 6,7,8 }

=== for each [p], (split [p] w.r.t [6],'b')

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

before split, partitions: { 0 1 2 3 4 5 },{ 6 7 8 }

new split of [0] is [1]

after split, partitions: { 0 2 3 4 5 }, { 1 },{ 6 7 8 }

before L: 0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 1 0 0

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

p and r are the new representatives. Now update L with the smallest of [0],[1] using [r] = [1],L[r]=C.size();

affter L: 0 1 2 3 4 5 6 7 8 0 2 0 0 0 0 1 0 0

===split[6] w.r.t (index of L)[6],'b') before split, partitions: 0 2 3 4 5 1 6 7 8

new split of [6] is [8] after split, partitions: 0 2 3 4 5 1 6 7 8

before L: 0 1 2 3 4 5 6 7 8 0 2 0 0 0 0 1 0 0

p and r are the new representatives. Now update L with the smallest of [6],[8] using [r] = [8],L[r]=C.size();

affter L: 0 1 2 3 4 5 6 7 8 0 2 0 0 0 0 1 0 2

— while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 2 0 0 0 0 1 0 2 Pick one [q] in L, Processing [q]= index of L = [1]，'b' current all partitions(eq.classes) repr: 0 1 6 8 current all partitions: 0 2 3 4 5 1 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[1],'b') ===split[0] w.r.t (index of L)[1],'b') before split, partitions: StateEqRel 0 2 3 4 5 1 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[1],'b') before split, partitions: 0 2 3 4 5 1 6 7 8

new split of [1] is [-1] ===split[6] w.r.t (index of L)[1],'b') before split, partitions: 0 2 3 4 5 1 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[1],'b') before split, partitions: StateEqRel 0 2 3 4 5 1 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 1 0 0 0 0 1 0 2 Pick one [q] in L, Processing [q]= index of L = [1], 'a' current all partitions(eq.classes) repr: 0 1 6 8 current all partitions: 0 2 3 4 5 1 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[1],'a') ===split[0] w.r.t (index of L)[1],'a') before split, partitions: 0 2 3 4 5 1 6 7 8

new split of [0] is [2] after split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

before L: L: 0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 1 0 2 p and r are the new representatives. Now update L with the smallest of [0],[2] using [p] = [0],L[r]=L[p]; L[p]=C.size(); affter 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 (index of L)[1],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[6] w.r.t (index of L)[1],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[1],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 2 0 0 0 0 0 1 0 2 Pick one [q] in L, Processing [q]= index of L = [0], 'b' current all partitions(eq.classes) repr: 0 1 2 6 8 current all partitions: StateEqRel 0 1 2 3 4 5 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[0],'b') ===split[0] w.r.t (index of L)[0],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[0],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[2] w.r.t (index of L)[0],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [2] is [-1] ===split[6] w.r.t (index of L)[0],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[0],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 1 0 0 0 0 0 1 0 2 Pick one [q] in L, Processing [q]= index of L = [0], 'a' current all partitions(eq.classes) repr: 0 1 2 6 8 current all partitions: StateEqRel 0 1 2 3 4 5 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[0],'a') ===split[0] w.r.t (index of L)[0],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[0],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[2] w.r.t (index of L)[0],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [2] is [-1] ===split[6] w.r.t (index of L)[0],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[0],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 1 0 2 Pick one [q] in L, Processing [q]= index of L = [6], 'a' current all partitions(eq.classes) repr: 0 1 2 6 8 current all partitions: StateEqRel 0 1 2 3 4 5 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[6],'a') ===split[0] w.r.t (index of L)[6],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[6],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[2] w.r.t (index of L)[6],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [2] is [4] after split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

before L: L: 0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0 2 p and r are the new representatives. Now update L with the smallest of [2],[4] using [p] = [2],L[r]=L[p]; L[p]=C.size(); affter 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 (index of L)[6],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[6],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 0 2 0 0 0 0 0 2 Pick one [q] in L, Processing [q]= index of L = [2], 'b' current all partitions(eq.classes) repr: 0 1 2 4 6 8 current all partitions: StateEqRel 0 1 2 3 4 5 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[2],'b') ===split[0] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[2] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [2] is [-1] ===split[4] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [4] is [-1] ===split[6] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 0 1 0 0 0 0 0 2 Pick one [q] in L, Processing [q]= index of L = [2], 'a' current all partitions(eq.classes) repr: 0 1 2 4 6 8 current all partitions: StateEqRel 0 1 2 3 4 5 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[2],'a') ===split[0] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[2] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [2] is [3] after split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

before L: L: 0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0 2 p and r are the new representatives. Now update L with the smallest of [2],[3] using [p] = [2],L[r]=L[p]; L[p]=C.size(); affter L: L: 0 1 2 3 4 5 6 7 8 0 0 2 0 0 0 0 0 2 胡 ===split[4] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [4] is [-1] ===split[6] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[2],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 0 1 0 0 0 0 0 2 Pick one [q] in L, Processing [q]= index of L = [2]，'a' current all partitions(eq.classes) repr: 0 1 2 3 4 6 8 current all partitions: StateEqRel 0 1 2 3 4 5 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[2],'a') ===split[0] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[2] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [2] is [-1] ===split[3] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [3] is [-1] ===split[4] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [4] is [-1] ===split[6] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[2],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0 2 Pick one [q] in L, Processing [q]= index of L = [8]，'b' current all partitions(eq.classes) repr: 0 1 2 3 4 6 8 current all partitions: StateEqRel 0 1 2 3 4 5 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[8],'b') ===split[0] w.r.t (index of L)[8],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[8],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[2] w.r.t (index of L)[8],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [2] is [-1] ===split[3] w.r.t (index of L)[8],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [3] is [-1] ===split[4] w.r.t (index of L)[8],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8 new split of [4] is [-1] ===split[6] w.r.t (index of L)[8],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[8],'b') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0 1 Pick one [q] in L, Processing [q]= index of L = [8]，'a' current all partitions(eq.classes) repr: 0 1 2 3 4 6 8 current all partitions: StateEqRel 0 1 2 3 4 5 6 7 8

=== for each [p], (split [p] w.r.t (index of L)[8],'a') ===split[0] w.r.t (index of L)[8],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [0] is [-1] ===split[1] w.r.t (index of L)[8],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [1] is [-1] ===split[2] w.r.t (index of L)[8],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [2] is [-1] ===split[3] w.r.t (index of L)[8],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [3] is [-1] ===split[4] w.r.t (index of L)[8],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [4] is [-1] ===split[6] w.r.t (index of L)[8],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [6] is [-1] ===split[8] w.r.t (index of L)[8],'a') before split, partitions: StateEqRel 0 1 2 3 4 5 6 7 8

new split of [8] is [-1] — while each [q], (split [p] w.r.t ([index of L]=[q],a)) L: 0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0 0

1.1 Minimization by equivalence of states

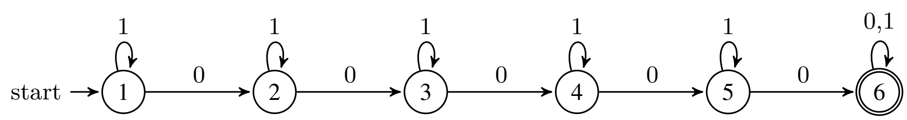


图 1.2: Minimizing example

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.

{a,b},{d,e}is not equivalent states.

Sets of equivalent states: {a,c},{b},{d},{e}

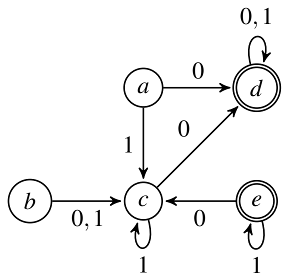


图 1.3: Finite state automaton

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.