

Лабораторная работа № 3. Минимизация конечного автомата

Задача минимизации конечного автомата, распознающего заданный язык, разрешима конструктивным методом.

Алгоритм минимизации конечного автомата можно определить в следующем порядке:

- 1) поиск и удаление всех недостижимых состояний;
- 2) поиск такого разбиения множества состояний автомата, при котором каждое подмножество содержит неразличимые состояния, т. е. если s и t принадлежат некоторому подмножеству, то все a из S $d(s, a)$ и $d(t, a)$ также принадлежат этому подмножеству. Для этого мы множество состояний разбивают на два подмножества: F и $S - F$;
- 3) попытка разбиения каждого из подмножеств с соблюдением указанного выше условия. Если возникает ситуация, при которой не удастся разбить никакое множество состояний, то процесс разбиения заканчивается;
- 4) в результате должен получиться некоторый набор множеств состояний $S_1 \dots S_k$, каждое из которых содержит только неразличимые состояния;
- 5) внесение в множество состояний минимизированного автомата по одному представителю каждого из множеств S_i . На этом процесс минимизации завершается.

Алгоритм минимизации целесообразно использовать и при распознавании эквивалентности двух заданных конечных автоматов.

Если необходимо выяснить, эквивалентны ли автоматы M_1 и M_2 , то достаточно минимизировать каждый из них. Если минимальные автоматы M'_1 и M'_2 имеют множества состояний с разным числом вершин, то исходные автоматы заведомо не эквивалентны.

Конечный автомат $M = (Q, T, F, H, Z)$ может содержать лишние состояния двух типов: недостижимые и эквивалентные состояния. Конечный автомат, не содержащий недостижимых и эквивалентных состояний, называется приведенным или минимальным конечным автоматом.

Одним из способов минимизации конечного автомата является устранение его недостижимых состояний, которое может быть выполнено с использованием одного из двух алгоритмов: устранения недостижимых состояний или объединения эквивалентных состояний.

Устранение недостижимых состояний автомата выполняется в следующем порядке.

1) поместить начальное состояние автомата в список достижимых состояний Q_d , т. е. $Q_d 0 = N$;

2) пополнить список группы достижимых состояний группой состояний \square приемников, отсутствующих в этом списке;

3) повторить п. 2, пока список достижимых состояний не перестанет меняться;

4) исключить из множества состояний конечного автомата все состояния, отсутствующие в списке достижимых состояний.

Порядок выполнения работы. Разработать программное средство, реализующее следующие функции:

- ввод исходного конечного автомата и вывод на экран его графа;
- устранение недостижимых состояний конечного автомата;
- исключение эквивалентных состояний конечного автомата;
- вывод на экран графа минимального конечного автомата.

Варианты индивидуальных заданий к лабораторной работе представлены в таблице

Вариант	Схема автомата	Вариант	Схема автомата	Вариант	Схема автомата
1		5		9	
2		6		10	
3		7		11	
4		8		12	

Описание алгоритма построения отношения эквивалентности на множестве состояний конечного автомата (этап I) и классов эквивалентных состояний в этом отношении (этап II)

Подразумевается, что в алгоритме используются операции над множествами, представимыми статическими массивами из целых, пар целых, то есть элементов вида $\text{index} = \text{struct} (\text{int } p, q)$ или элементов вида $\text{class} = [] \text{int}$, то есть одномерных массивов целых с разным числом элементов, но не больше, чем число состояний автомата.

Над значениями вида $index$ определена операция выборки поля (см. строки 47–49 в описании алгоритма). Для пополнения таких множеств новыми элементами используются операции U соответствующего вида.

При пополнении массивов новыми элементами с помощью присваивания предполагается, что значение правого операнда присваивается левому только в том случае, когда он не равен ни одному элементу массива, представляющего множество соответствующего вида (см. строки 10, 21, 30, 53 в алгоритме).

Формулы вида $]S$, где S обозначает множество (массив), дают число S элементов, значения которых определены. В алгоритме используются следующие обозначения и представления:

1. Конечный автомат представляется матрицей переходов, строки которой индексируются номерами состояний, а столбцы — входными символами $a \in \Sigma$.

2. Состояние автомата представляется его номером $1 \leq i \leq n$, где $n =]Q$ — число состояний, причём состояние 1 считается начальным.

3. Принимая во внимание рефлексивность и симметричность отношения эквивалентности \mathcal{R} , оно ради экономии памяти представляется статическим массивом $E[1 : e]_{hypo}$, элементами которого являются пары номеров состояний (p, q) , где $p < q$, называемые гипотезами. Гипотеза представляется как значение вида $hypo$. Число элементов в массиве E оценивается по формуле $e = (n(n+1)) \div 2$.

4. Понятие подобности состояний переносится также на гипотезы $h = (p, q)$ с помощью формулы $\approx h$, которая реализует проверку подобности состояний, составляющих гипотезу h .

5. $H[1 :]\Sigma]$ — массив гипотез, относящихся к парам подобных состояний.

6. k — индекс текущей (опорной) гипотезы в массиве H . Оператор $H[k+ := 1] := h$ пополняет множество гипотез в том случае, когда h ещё не находится в массиве H .

7. $CLASS[1 : n]_{int}$ — элементы текущего класса состояний.

8. $CLASS INDEX[1 : n]_{class}$ — индекс классов. Элемент этого массива есть множество состояний исходного автомата, составляющих один класс эквивалентности.

9. $B[1 : n]_{int}$ — множество базовых элементов класса.

10. $A[1 : n]_{int}$ — множество состояний, включённых во все классы эквивалентности.

Алгоритм. Построение классов эквивалентности в отношении \mathcal{R} .

Вход: Приведённый dfa $M = (Q, \Sigma, \delta, q_0, F)$.

Выход: CLASS INDEX — множество классов эквивалентности в отношении \mathcal{R} .

Метод: 1.

```
1.      begin {ЭТАП I: построение отношения эквивалентности на множестве состояний}
2.          m := 0; { Множество пар эквивалентных состояний (p,q), где p < q }
3.          for i to n-1 {1 ≤ i ≤ n, где n = |Q|}
4.              do { i }
5.              for j from i+1 to n { j ≤ i+1 ≤ n }
6.                  do { j } h := (i, j); { Очередная пара состояний: i < j }
7.                  k := 0; {Число элементов в массиве H }
8.                  if (h ∉ E) ∧ (≈ h) {Состояния i и j подобны}
9.                      then { Гипотезы h в E нет. Инициализация массива гипотез: }
10.                     H[k+ := 1] := h; {В H теперь только одна гипотеза}
11.                     Continue := true;
12. { Цикл проверки гипотез }
13.      for l while (l ≤ k) ∧ Continue
14.          do h := H[l] { Перебор гипотез}
15.          for ∀ ∈ D(h) { Цикл пополнения гипотез }
16.              do h0 := (δ(p of h, a), δ(q of h, a));
17.              if ≈ h0 { Гипотеза касается состояний }
18.                  then { с одинаковыми областями действия }
20.                  if h0 ∉ H[1 : k]
21.                      then H[k+ := 1] := h0 { H пополняется h0 }
22.                      fi 2
23.                  fi
24.              od { Конец цикла пополнения гипотез}
25.          od; { Конец цикла проверки гипотез }
26.      for l to k
27.          do
```

```

28.      h := H[l] ;
29.      if h ∉ E[1 : m]
30.      then E[m+ := 1] := h
31.      fi
32.      od;
33. { Текущий этап пополнения массива E проверенными гипотезами закончен }
34.      Continue := false
35.      fi
36.      od {}
37.      od {} ; { ЭТАП II: построение классов эквивалентности состояний }
39.      if m > 0 { E, ∅ }
40.      then
41.      A := ∅; { Множество задействованных состояний в
классов эквивалентности }
42.      B := {1}; { Инициализация начального класса }
43.      i := 0; { Инициализация индекса классов }
44.      next : { Цикл построения очередного класса }
45.      for ∀h : (h ∈ E)
46.      do
47.      if (p of h) ∈ B
48.      then B := B ∪ (q of h)
49.      elif q of h ∈ B then B := B ∪ (p of h)
50.      fi
51.      od;
52.      C L A S S := B; { Очередной класс }
53.      C L A S S I N D E X[i+ := 1] := C L A S S; { Фиксация класса в индексе классов }
54.      A := A ∪ B; { Фиксация состояний, включённых в классы эквивалентности }
55.      if ]A < ]Q
56.      then { Не все состояния распределены по классам }
57.      Continue = true;

```

```
58.      for  $\forall b : (b \in Q)$  while Continue
59.          do { Цикл нахождения базового элемента следующего класса }
60.          if  $b \notin A$ 
61.              then  $B := b$ ; Continue := false
62.              fi
63.          od;
64.          goto next { Возврат на построение следующего класса }
65.          fi
66.          fi
67.  end.
```