Компьютерные инструменты в образовании, 2017
№ 1: 5–14
УДК: 519.682.1 + 681.142.2
http://ipo.spb.ru/journal

информатика

ЕЩЁ ОДИН МЕТОД МИНИМИЗАЦИИ КОНЕЧНЫХ АВТОМАТОВ

Мартыненко Б.К.[1]

[1]СПбГУ, Санкт-Петербург, Россия

Аннотация

Описывается алгоритм минимизации приведённых детерминированных конечных автоматов, основанный на использовании классов эквивалентности по признаку неразличимости множеств цепочек, принимаемых в соответствующих состояниях. Проводится сравнение с алгоритмом Дж. Хопкрофта, в котором классы эквивалентных состояний строятся, исходя из отношения различимости состояний по допустимым входным цепочкам.

Ключевые слова: *конечный автомат, диаграмма состояний, неразличимые состояния, алгоритм Дж.Хопкрофта.*

Цитирование: Мартыненко Б.К. Ещё один метод минимизации конечных автоматов // Компьютерные инструменты в образовании. 2017. № 1. С. 5–14.

ВВЕДЕНИЕ

Метод минимизации конечных автоматов, описываемый в данной статье, появился в процессе продолжения работ по совершенствованию этапа видонезависимого синтаксического анализа в проекте реализации алгоритмического языка Алгол 68 [1–3] в Ленинградском государственном университете в 1969–1975 годах.

Г.С. Цейтин, научный руководитель проекта, предложил модель синтаксического анализа, основанную на автоматных грамматиках и скобочных структурах. Отправной точкой при разработке метода послужила простая модель: регулярные выражения как средство описания языка и конечные автоматы в качестве адекватного средства его распознавания.

Конечные автоматы часто используют в качестве драйверов для преобразования регулярных языков [4], выполняемых во время сканирования входного текста. Логично было бы найти метод получения конечного автомата с минимальным числом состояний для языка, заданного регулярным выражением, не забывая, однако, что минимизация сужает семантический контекст вышеупомянутых преобразований.

Две теоремы из [5] дают ключ к решению проблемы минимизации конечных детерминированных автоматов.

Во-первых, Теорема 3.1 утверждает, что если отношение эквивалентности R, определённое через язык $L \subseteq \Sigma^*$ следующим образом: $(\forall x, y \in \Sigma^*) : xRy \Leftrightarrow (\forall z \in \Sigma^*) : (xz \in L \Leftrightarrow yz \in L)$ имеет конечный индекс[1], то язык L принимается некоторым dfa[2]. Доказательство

[1] Индекс отношения эквивалентности — это мощность множества классов эквивалентности.

[2] Dererministic finite automata (детерминированный конечный автомат).

этой теоремы конструктивно: строится dfa M, в роли состояний которого используются классы эквивалентности отношения R.

Во-вторых, в Теореме 3.2 утверждается, что этот конечный автомат M является минимальным по числу состояний.

Идея состоит в том, чтобы перенести отношение эквивалентности R, определённое на цепочках в алфавите регулярного языка L, на аналогичное отношение эквивалентности $\Re$, определённое на состояниях автомата, а затем использовать классы эквивалентности $\Re$ вместо состояний исходного автомата. Именно, два состояния p и q эквивалентны в отношении $\Re$, если множества цепочек во входном алфавите автомата, принимаемых в этих состояниях, одинаковы (*см*. опр. 5, разд. 1).

Итак, как следствие вышеупомянутых теорем, заключаем, что конечный автомат с минимальным числом состояний может быть получен в три этапа:

1) построение отношения эквивалентности $\Re$ на множестве состояний данного автомата, исходя из неразличимости цепочек, принимаемых в соответствующих состояниях,

2) построение классов эквивалентности в отношении $\Re$,

3) использование классов эквивалентности отношения $\Re$ взамен состояний данного автомата.

В Разделе 1 описывается алгоритм построения отношения $\Re$ и классов эквивалентности в этом отношении. Он реконструирован из программы ТК SYNTAX, написанной в конце 70-х на языке Алгол 68, в частности, с целью тестирования первой версии компилятора.

В Разделе 2 описывается метод Хопкрофта, в котором классы эквивалентных состояний строятся, исходя из другого отношения $\equiv$, основанного на *различимости* множеств цепочек, принимаемых в соответствующих состояниях.

В Разделе 3 оба метода сравниваются на примере регулярного языка, распознаваемого конечным автоматом, позаимствованным из [6, ch. 2, p. 45–46]. Этот язык определяется регулярным выражением. По нему строится конечный автомат. Он, как оказывается, не является минимальным по числу состояний. Однако оба метода минимизации этого автомата при использовании этих двух разных отношений дают один и тот же результат.

1. АЛЬТЕРНАТИВНЫЙ МЕТОД МИНИМИЗАЦИИ ЧИСЛА СОСТОЯНИЙ

Считая, что основные понятия теории формальных языков и автоматов известны, определим только необходимые.

Определение 1. *Детерминированный конечный автомат (dfa)* есть формальная система $M = (Q, \Sigma, \delta, q_0, F)$, где Q — конечное множество состояний, Σ — входной алфавит, $\delta : Q \to \Sigma \times Q$ — функция переходов, q_0 — начальное состояние, $F \subseteq Q$ — множество конечных состояний.

Когда функция переходов всюду определена, говорят, что автомат *полный*.

Слово (*цепочка*) есть любая конечная последовательность символов $a \in \Sigma$.

Пусть Σ^* обозначает множество слов над алфавитом Σ, а ε — *пустое слово*.

Определим *расширенную функцию переходов* $\widehat{\delta} : Q \to \Sigma^* \times Q$ следующим образом:

$$\widehat{\delta}(q, \varepsilon) = q, \quad \widehat{\delta}(q, xa) = \delta(\widehat{\delta}(q, x), a).$$

Определение 2. *Язык T(M), допускаемый (принимаемый, распознаваемый) конечным автоматом M*, есть множество слов $w \in \Sigma^*$, таких что $\widehat{\delta}(q_0, w) \in F$.

Определение 3. *Dfa M* будем называть *приведённым*, если

$$(1)\ \forall q \in Q : \exists x \in \Sigma^* : \delta(q_0, x) = q, \quad (2)\ \forall p \in F : \exists y \in \Sigma^* : \delta(q_0, y) = p.$$

Термин «приведённый» в применении к автомату аналогичен этому термину в применении к грамматикам Н. Хомского, в том смысле, что все нетерминальные и терминальные символы используются при порождении языка и только они. В применении к конечному автомату это означает, что всё множество цепочек, принимаемых автоматом, задействует все его состояния и входные символы и только их.

Однако это не значит, что не может существовать другая грамматика типа 3 с меньшим числом нетерминалов, порождающая тот же самый язык. Аналогично не исключено, что данный регулярный язык может приниматься другим автоматом с меньшим числом состояний.

Определение 4. Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — приведённый *dfa*. Состояния $p, q \in Q$ будем называть *подобными* и писать $p \approx q$, если

$$((p \in F \wedge q \in F) \vee (p \notin F \wedge q \notin F)) \wedge (D(p) = D(q)),$$

где $D(p),\ D(q)$ — множества допустимых входных символов $a \in \Sigma$ в состоянии p и q соответственно.

Определение 5. Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — приведённый *dfa*, и $T(M) = L$. Определим отношение эквивалентности $\Re$ на множестве состояний Q следующим образом:

$$p \Re q \Leftrightarrow (D(p) = D(q)) \wedge (\forall a \in D(p) : \delta(p, a)\ \Re\ \delta(q, a)).$$

Далее даётся описание алгоритма построения отношения эквивалентности на множестве состояний конечного автомата (этап I) и классов эквивалентных состояний в этом отношении (этап II) в стиле программы на языке типа Алгол 68 с комментариями. Подразумевается, что в алгоритме используются операции над множествами, представимыми статическими массивами из целых, пар целых, то есть элементов вида **index** = **struct** (**int** p, q) или элементов вида **class** = [ ] **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.* строчки 47–49 в описании Алгоритма 1).

Для пополнения таких множеств новыми элементами используются операции $\cup$ 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.* строчки 10, 21, 30, 53 в Алгоритме 1).

Формулы вида $\sharp S$, где S обозначает множество (массив), дают число S элементов, значения которых определены.

В Алгоритме 1 используются следующие обозначения и представления:

1. Конечный автомат представляется матрицей переходов, строчки которой индексируются номерами состояний, а столбцы — входными символами $a \in \Sigma$.
2. Состояние автомата представляется его номером $1 \leqslant i \leqslant n$, где $n = \sharp Q$ — число состояний, причём состояние 1 считается начальным.
3. Принимая во внимание рефлексивность и симметричность отношения эквивалентности $\Re$, оно ради экономии памяти представляется статическим массивом $E[1 : e]$**hipo**, элементами которого являются пары номеров состояний (p, q), где

$p < q$, называемые *гипотезами*. Гипотеза представляется как значение вида **hipo**. 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. $H[1 : \sharp\Sigma]$ — 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 1. Построение классов эквивалентности в отношении $\Re$.

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

Выход: CLASS INDEX — множество классов эквивалентности в отношении $\Re$.

Метод:

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 {$1 \le i \le n$, где $n = \sharp Q$ }
4. **do** { i }
5. **for** j **from** $i + 1$ **to** n {$j \le i + 1 \le n$}
6. **do** { j } $h := (i, j)$; { Очередная пара состояний: $i < j$ }
7. $k := 0$; {Число элементов в массиве H }
8. **if** $(h \notin E) \wedge (\approx h)$ {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** I **while** $(I \le k) \wedge Continue$
14. **do** $h := H[I]$ { Перебор гипотез}
15. **for** $\forall \in D(h)$ { Цикл пополнения гипотез }
16. **do** $h' := (\delta($ p **of** $h, a),\ \delta($q **of** $h, a))$;
17. **if** $\approx h'$ { Гипотеза касается состояний }
18. **then** { с одинаковыми областями действия }
20. **if** $h' \notin H[1 : k]$
21. **then** $H[k+ := 1] := h'$ { Н пополняется h' }
22. **fi**
23. **fi**
24. **od** { Конец цикла пополнения гипотез}
25. **od**; { Конец цикла проверки гипотез }
26. **for** I **to** k
27. **do**
28. $h := H[I]$;

```
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 {j}
37. od {i};
    { ЭТАП II: построение классов эквивалентности состояний }
39. if m > 0 { E ≠ ∅ }
40. then
41.     A := ∅; { Множество задействованных состояний в классах 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.     CLASS := B; { Очередной класс }
53.     CLASS INDEX[i+ := 1] := CLASS; { 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 ∀b : (b ∈ Q) while Continue
59.         do { Цикл нахождения базового элемента следующего класса }
60.             if b ∉ 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
```

Аналог отношения эквивалентности $\Re$ представлен массивом E. Если при окончании этапа I оказывается, что $m = 0$, то исходный автомат уже минимальный: каждый класс эквивалентности содержит по одному состоянию исходного автомата. В противном случае массив *CLASS INDEX* представляет классы состояний в отношении эквивалентности $\Re$. Подставляя эти классы вместо состояний исходного автомата, мы получаем минимальный автомат, принимающий тот же самый язык.

2. МЕТОД ХОПКРОФТА ПОСТРОЕНИЯ КАНОНИЧЕСКОГО КОНЕЧНОГО АВТОМАТА

Напомним несколько определений и фактов из [7], необходимых для сравнения метода Дж. Хопкрофта и метода, предлагаемого в данной статье, применительно к минимизации детерминированных конечных автоматов.

Определение 6. Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — конечный автомат, $q_1, q_2 \in Q$, и $q1 \neq q2$. Считается, что $x \in \Sigma^*$ *различает* q_1 и q_2, если $(q_1, x) \vdash^* (q_3, \varepsilon), (q_2, x) \vdash^* (q_4, \varepsilon)$ и только одно из q_3 и q_4 принадлежит F.

Говорят, что q_1 и q_2 *k-неразличимы*, и пишут $q_2 \equiv^k q_2$, если и только если не существует никакого $x \in \Sigma^*$, такого что $| x | \leq k$, который бы различал q_1 и q_2.

Говорят, что состояния q_1 и q_2 неразличимы по Хопкрофту, и пишут $q_1 \equiv q_2$, если и только если они k-неразличимы для всех $k \geq 0$.

Определение 7. Конечный автомат M будем считать приведённым по Хопкрофту, если все состояния достижимы по п. (1) определения 3 и никакие два состояния $q_1 \neq q_2$ не различаются по Хопкрофту.

Поскольку отношение неразличимости состояний по Хопкрофту $\equiv$ равнозначно отношению эквивалентности $\Re$ по определению 5, то приведённость конечного автомата по Хопкрофту равнозначна тому, что он минимален по числу состояний.

Алгоритм Хопкрофта [7] (*см.* Алгоритм 2.2 ниже) построения такого автомата с минимальным числом состояний опирается на следующую лемму:

Лемма 2.11. *Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — конечный автомат с n состояниями. Состояния q_1 и q_2 неразличимы тогда и только тогда, когда они $(n–2)$-неразличимы.*

При её доказательстве отмечается, что $q_1 \equiv^k q_2$ при двух следующих условиях:

(1) $q_1 \equiv^0 q_2$ тогда и только тогда, когда q_1 и q_2 оба либо принадлежат, либо не принадлежат F,

и

(2) $q_1 \equiv^k q_2$ тогда и только тогда, когда $q_1 \equiv^{k-1} q_2$ и $\delta(q_1, a) \equiv^{k-1} \delta(q_2, a)$ для всех $a \in \Sigma$, из чего следует, что $\equiv \supseteq \equiv^{n-2} \equiv \supseteq \equiv^{n-3} \ldots \supseteq \equiv^2 \supseteq \equiv^1 \supseteq \equiv^0$.

Очевидно, что приведённость конечного автомата по Хопкрофту равнозначна тому, что он минимален по числу состояний.

Алгоритм 2.2. (Хопкрофт). Построение канонического конечного автомата.

Вход: Конечный автомат $M = (Q, \Sigma, \delta, q_0, F)$.

Выход: M' — приведённый по Хопкрофту (*см.* опр. 7) конечный автомат, эквивалентный автомату M.

Метод:

Шаг 1: Исключить из автомата M все состояния, не достижимые из q_0.

Шаг 2: Строить отношения эквивалентности $\equiv^0$, $\equiv^1$, $\ldots$, как описано в лемме 2.11, до тех пор, пока не совпадут $\equiv^{k+1} = \equiv^k$ при некотором k. Взять в качестве $\equiv$ отношение $\equiv^k$.

Шаг 3: Построить конечный автомат $M' = (Q', \Sigma, \delta', q_0', F')$, где

3.1 Q' — множество классов эквивалентности отношения $\equiv$, где $[p]$ — класс эквивалентности отношения $\equiv$, содержащий состояние p;

3.2 $\delta'([p], a) = [q]$, если $\delta(p, a) = q$;

3.3 $q_0' = [q_0]$;

3.4 $F' = \{[q] | q \in F\}$. □

Можно непосредственно доказать, что шаг 3.2 непротиворечив, то есть какой элемент класса $[p]$ ни взять, значение $\delta'([p], a)$ будет одним и тем же классом.

Доказательство равенства $L(M) = L(M')$ просто. Его можно найти во многих учебниках[3]. Остаётся убедиться в том, что автомат с меньшим числом состояний, чем у M', не может допускать $L(M)$. Теорема 2.6 из [7] даёт убедительный ответ на справедливость этого утверждения.

Теорема 2.6. Автомат M', который строится алгоритмом 2.2, имеет наименьшее число состояний среди всех конечных автоматов, допускающих язык $L(M)$.

Доказательство

Предположим, что M'' имеет меньше состояний, чем M', и что $L(M'') = L(M)$. В силу шага 1 алгоритма 2.2 каждое состояние M' достижимо. Так как M'' имеет меньше состояний, чем M', то найдутся цепочки w и x, переводящие состояние q_0' в разные состояния, а q_0'' (начальное состояние автомата M'') в одно и то же:

$$(q_0'',\ w) \vdash_{M''}^{*} (q,\ \varepsilon) \text{ и } (q_0'',\ x) \vdash_{M''}^{*} (q,\ \varepsilon).$$

Следовательно, w и x переводят автомат M в различные состояния, скажем, p и r. Это значит, что существует цепочка y, такая что только одна из цепочек wy и xy принадлежит $L(M)$. Но wy и xy должны переводить M'' в одно и то же состояние s, для которого $(q, y) \vdash^{*} (s, \varepsilon)$. Таким образом, точно одна из цепочек wy и xy не может принадлежать $L(M'')$, а это противоречит предположению о том, что $L(M'') = L(M)$. □

3. ПРИМЕР: СРАВНЕНИЕ С МЕТОДОМ ХОПКРОФТА

С целью сравнения Алгоритма 1 (*см.* раздел 1) с Алгоритмом 2.2 Хопкрофта [7], выполним его на примере языка, распознаваемого автоматом A_0, позаимствованного из [6, ch. 2, p. 45–46] (*см.* рис. 1 ниже).

Автомат A_0 читает цепочки символов 0 и l и распознаёт их как двоичные числа, конгруэнтные 2 (по модулю 3). Обозначение вида $v_3(x)$ представляет бинарную цепочку x как целое неотрицательное значение по модулю 3.

Например, $v_3(100) = 1$ и $v_3(1011) = 2$.

Рассмотрим произвольную входную цепочку $w = a_1 \dots a_n$ на входе A_0, где каждый $a_i (1 \le i \le n)$ есть или 0 или 1. Ясно, что для каждого i цепочка $a_1 \dots a_i$ попадает в один из этих трёх случаев:

(0) $v_3(a_1 \dots a_i) = 0$,
(1) $v_3(a_1 \dots a_i) = 1$,
(2) $v_3(a_1 \dots a_i) = 2$.

Никакие другие случаи невозможны. Так что A_0 нуждается только в трёх состояниях, которые соответствуют вышеупомянутым трём случаям. Обозначим эти три состояния p_0, p_1 и p_2 соответственно. Состояние p_0, соответствующее случаю (0), является стартовым, состояние p_1 соответствует случаю (1), состояние p_2, соответствующее случаю (2), является конечным. Правила, которые управляют изменениями состояний, должны быть определены соответственно.

Заметим, что $v_3(a_1 \dots a_{i+1}) = 2 \times v_3(a_1 \dots a_i) + a_{i+1} \pmod 3$. Так, если текущее состояние есть p_1 и текущий входной символ есть 1, то следующее состояние есть p_0, поскольку $2 * 1 + 1 = 0 \pmod 3$. Ясно, что каждый шаг изменения состояния единственным образом определён текущим состоянием и текущим входным символом. Мы выделяем состояние

[3] *См.*, например, [5].

p_2 как конечное состояние и определяем, что A_0 принимает вход w, если он находится в состоянии p_2 после чтения последнего символа w. Очевидно, что dfa A_0 является минимальным по числу состояний. Он представлен на рис. 1.

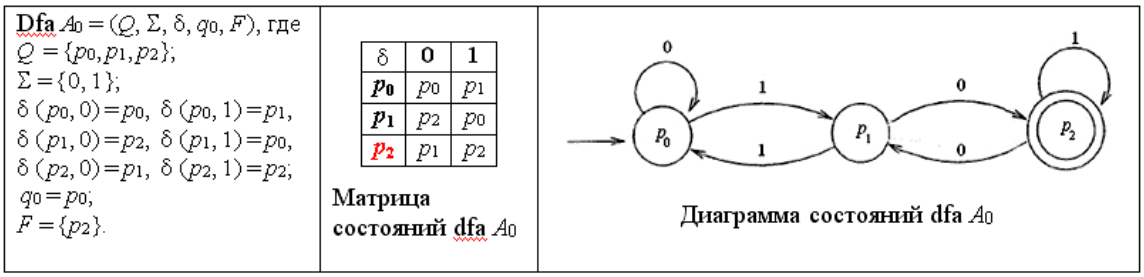

Рис. 1. Минимальный dfa A_0

Теперь мы проделаем следующий эксперимент в три этапа.

1. Построим регулярное выражение для языка, распознаваемого dfa A_0:

$$(0 \mid 1\ \ 1)^*\ \ 1\ \ 0\ \ (1 \mid 0\,(1\ \ 0^*\ \ 1)^*\ \ 0)^* \qquad (1)$$

2. По этому регулярному выражению построим dfa A_1 (*см.* табл. 1), который имеет большее число состояний, чем dfa A_2 (*см.* табл. 2).

3. Используя вышеописанный алгоритм 1 (этап 1) получения отношения $\Re$ на состояниях A_1, построим dfa A_2, заменяя состояния A_1 на классы эквивалентности этого отношения. Получаем dfa A_2, который, как видим, равен автомату dfa A_0.

Таблица 1. Матрица переходов dfa A_1

Состояние	0	1
1	2	3
2	2	3
3	4	5
4 – конечное	6	7
5	2	3
6	8	9
7 – конечное	6	7
8 – конечное	6	7
9	10	11
10	10	11
11	8	9

Таблица 2. Матрица переходов dfa $A_2 = A_0$

Класс эквив. состояний	0	1
[1] = { 1, 2, 5, 9, 10 }	[1]	[2]
[2] = {3, 6, 11}	[3]	[1]
[3] = {4, 7, 8} – конечное	[2]	[3]

Начальное состояние автомата A_1 имеет номер 1.

Обратимся к Алгоритму 2.2 из [7] и продемонстрируем его выполнение на автомате A_1, построенном из регулярного выражения (1) (*см.* табл. 1).

Сначала вычислим отношение ≡, строя последовательность приближений $\equiv^k$, начиная с $k = 0$. Следуя Алгоритму 2.2, разобъём всё множество состояний данного автомата на два подмножества в соответствии с Леммой 2.11: {4, 7, 8} и {1, 2, 3, 5, 6, 9, 10, 11}. В первое включены все конечные состояния, во второе — все остальные. Они представляют начальное приближение классов эквивалентности отношения ≡. Ход разбиения множества состояний автомата A_1 на подмножества приближений к классам эквивалентности в отношении ≡ представлен в табл. 3.

Таблица 3

Степень уточнения	Уточнённые классы неразлчимых состояний
$\equiv^0$	{4, 7, 8}, {1, 2, 3, 5, 6, 9, 10, 11}
$\equiv^1$	{4, 7, 8}, {1, 2, 3, 5, 9, 10}, {3, 6, 11}
$\equiv^2$	{4, 7, 8}, {1, 2, 3, 5, 9, 10}, {3, 6, 11}

Итак, $\equiv^1=\equiv^2=\equiv$, и процесс нахождения классов эквивалентности в отношении $\equiv$ автомата A_1 закончен. Заметим, что множество классов эквивалентных состояний автомата A_1, полученное в отношении $\equiv$ при помощи Алгоритма 2.2, равно такому же множеству в отношении $\Re$ = {(1, 5), (1, 9), (1, 10), (2, 5), (2, 9), (2, 10), (3, 6), (3, 11), (4, 7), (5, 10), (6, 11), (7, 8), (9, 10)}, полученному посредством Алгоритма 1.

Замена состояний в матрице переходов автомата A_1 (табл. 1) на классы эквивалентности отношения $\equiv$ или $\Re$ даёт матрицу переходов автомата A_2 (табл. 2), равную матрице переходов автомата A_0 с минимальным числом состояний априори.

4. ЗАКЛЮЧЕНИЕ

Алгоритм 1 и Алгоритм 2.2 Дж. Хопкрофта [7] базируются на использовании классов эквивалентности по признаку неразличимости множеств цепочек, принимаемых в соответствующих состояниях соответственно.

Первый из этих алгоритмов непосредственно строит отношение как множество пар *неразличимых* состояний по движениям автомата, а классы эквивалентности получаются в результате транзитивного замыкания этого отношения.

Второй алгоритм, начиная с грубого разделения состояний на два подмножества, первое из которых включает все конечные состояния, а второе — все остальные состояния, и затем уточняет это разбиение путём перераспределения состояний по признаку *различения* соответствующих переходных состояний.

Иначе говоря, эти алгоритмы основываются на двойственных методах и могут иметь разные предпочтения при их реализации. Например, в случае, когда автомат минимальный или близкий к минимальному, то есть основание полагать, что Алгоритм 1 даст ответ быстрее Алгоритма 2.2, поскольку он получит результат уже по признаку подобия ($\approx$), который аналогичен признаку различия степени 0 ($\equiv^0$). Реализация этих двух алгоритмов имеет схожую оценку сложности порядка $kn \log n$, где k — некоторая константа, линейно зависимая от размера входного алфавита, а n — число состояний конечного автомата [8].

Список литературы

1. Пересмотренное сообщение об Алголе 68. / Ред. А. Ван Вейнгаарден. М.: Мир, 1979.
2. Алгол 68. Методы реализации / Ред. Г.С. Цейтин. Л.: Изд-во Ленингр. ун-та, 1976.
3. *Мартыненко Б.К.* Синтаксически управляемая обработка данных. СПб.: Изд-во СПб ун-та, 2004.
4. *Мартыненко Б.К.* Челночные трансляции в SYNTAX-технологии // Компьютерные инструменты в образовании. 2014. № 5. С. 3–15.
5. *Hopcroft J.E., Ullman J.D.* Formal languages and their relation to automata. Reading, MA: Addison-Wesley Pub. Co., Inc., 1969.
6. *Rozenberg G., Salomaa A.* Handbook of Formal Languages. Vol. 1: Word, Language, Grammar. Berlin, Heidelberg: Springer-Verlag, 1997.

7. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. Т. 1. М.: Мир, 1978.
8. *Hopcroft J.E.* An n log n algorithm for minimizing ststes in a finite automaton. Technical Report CS-71-190, Stanford University, January 1971.

Поступила в редакцию 10.01.2017, окончательный вариант — 12.02.2017.

Computer tools in education, 2017
№ 1: 5–14
http://ipo.spb.ru/journal

ONE MORE METHOD FOR MINIMIZATION OF FINITE AUTOMATA

Martynenko B.K.[1]

[1]SPBSU, Saint-Petersburg, Russia

Abstract

An algorithm for minimizing the number of states of the well-formed deterministic finite automaton is described. It is based on the use of equivalence classes on the relation of the indistinguishability of string sets accepted by the automaton. A comparison is made with the well-known algorithm of J.E. Hopcroft, which constructs the classes of equivalent states on the property of the distinguishability of input strings accepted in the corresponding states.

Keywords: *finite automaton, state diagram, indistinguishable states, Hopkroft's algorithm.*

Citation: Martynenko, B., 2017. "Eshche odin metod minimizatsii konechnykh avtomatov" ["One more method for minimization of finite automata"], *Computer tools in education*, no. 1, pp. 5–14.

Received 10.01.2017, the final version — 12.02.2017.

Boris K. Martynenko, Professor of Computer Sciences at Math.-Math. department of SPbSU, Universitetsky prospekt, 28, 198504, Saint Petersburg, Russia mbk@ctinet.ru

Мартыненко Борис Константинович, доктор физико-математических наук, профессор кафедры информатики математико-механического факультета СПбГУ; 198504, Россия, Санкт-Петербург, Старый Петергоф, Университетский пр., д. 28, математико-механический факультет, кафедра информатики, mbk@ctinet.ru