УДК 519.713

А.Н. Чеботарев

О минимизации автоматов алгоритмом Хопкрофта

Рассмотрен алгоритм Хопкрофта для минимизации детерминированных вполне определенных конечных автоматов. Даны понятные доказательства корректности алгоритма и оценки временной сложности. Доказательство основано на предложенном понятии дерева расщеплений и методе распространения «закрытых» вершин в этом дереве.

Ключевые слова: алгоритм Хопкрофта, минимизация автомата, разбиение, конгруэнция, временная сложность, дерево расщеплений.

Розглянуто алгоритм Хопкрофта для мінімізації детермінованих всюди визначених скінченних автоматів. Дано зрозуміле доведення коректності алгоритму та оцінки часової складності. Доведення базується на запропонованому понятті дерева розщеплень і методі розповсюдження «закритих» вузлів у цьому дереві.

Ключові слова: алгоритм Хопкрофта, мінімізація автомата, розбиття, конгруенція, часова складність, дерево розщеплень.

Введение. Минимизация конечных автоматов имеет существенное значение при решении задач, использующих автоматные модели, такие как синтаксический анализ текстов, проектирование дискретных систем управления и многие другие. Проблема минимизации вполне определенных автоматов впервые была сформулирована в 50-х годах прошлого века в работах Хаффмена [1] и Мура [2]. Муром был предложен подход к построению минимального автомата. С тех пор опубликовано много работ, описывающих и уточняющих алгоритм минимизации. Простые рассуждения показывают, что временная сложность этого алгоритма оценивается как $O(n^2)$ или точнее $O(kn^2)$, где k – мощность входного алфавита, а n – количество состояний автомата. Ввиду распространенности применения процедуры минимизации, естественно желание уменьшить сложность алгоритма. В 1971 г. Дж. Хопкрофт предложил алгоритм, сложность которого в худшем случае оценивается как $O(kn \log n)$. Этот результат до сих пор остается лучшим. Подход Хопкрофта, хотя идейно близок к подходу Мура, существенно отличается от него и, если последний воспринимается естественно и корректность его достаточно очевидна, то более изощренный метод минимизации Хопкрофта требует больших усилий для его осмысления и доказательства корректности.

В оригинальной работе Хопкрофта [3] было дано довольно сложное описание алгоритма и не очень строго обосновывалась его корректность и оценка временной сложности. Поэтому за ней последовал ряд работ, например [4–7], в которых алгоритм изложен более ясно и предложено более строгое и менее громоздкое обоснование его корректности и оценки временной сложности. Первая такая попытка была сделана Грисом [4] в 1973 г. В этой работе вопросы, связанные с обоснованием и анализом алгоритма Хопкрофта, изложены более строго. Однако и эта работа в дальнейшем вызвала некоторые нарекания. Кроме того в русскоязычной литературе практически отсутствует простое описание алгоритма Хопкрофта и, тем более, его обоснование.

Задачей настоящей статьи является достаточно простое и ясное изложение как самого алгоритма, так и доказательства его корректности и оценки временной сложности. С этой целью предложены новые (по мнению автора, более простые) доказательства утверждений, необходимых для обоснования корректности алгоритма. Обычно алгоритм Хопкрофта формулируется для минимизации автомата-распознавателя, в статье рассматриваются также его особенности при минимизации автоматов Мура и Мили. Следует заметить, что приведенные в литературе оценки сложности алгоритма справедливы при выполнении определенных требований к структурам данных, используемых при реализации алгоритма. Поэтому в статье приведены эти требования и описаны удовлетворяющие им структуры данных.

Основные понятия

Разбиением множества Q называется такое семейство $\pi = \{B_1, B_2, ..., B_n\}$ непустых, попар-

но непересекающихся подмножеств множества Q, что $Q = B_1 \cup B_2 \cup ... \cup B_n$. Подмножества B_i (i = 1, ..., n) называются *классами разбиения* π. Разбиение множества B на два непустых подмножества будем называть *расщеплением B*.

Понятие разбиения множества связано с понятием отношения эквивалентности на этом множестве так, что каждый класс разбиения совпадает с классом соответствующей эквивалентности. Таким образом, между разбиениями множества Q и эквивалентностями на нем имеется взаимно однозначное соответствие. *Произведением* разбиений π_1 и π_2 (обозначается $\pi_1 \wedge \pi_2$) называется разбиение π, классы которого представляют собой попарные пересечения классов π_1 с классами π_2. На множестве разбиений следующим образом определен частичный порядок: разбиение π_1 *уточняет* (*refines*) разбиение π_2 ($\pi_1 \le \pi_2$), если каждый класс разбиения π_1 содержится в некотором классе разбиения π_2. Частичный порядок на множестве разбиений определяет частичный порядок по включению ($\subseteq$) на множестве соответствующих эквивалентностей. Пусть разбиениям π_1, π_2 соответствуют эквивалентности θ_1, θ_2, тогда из $\pi_1 \le \pi_2$ следует $\theta_1 \subseteq \theta_2$.

Утверждение 1. Пусть θ_1 и θ_2 – эквивалентности на Q. Если для любых $a, b \in Q$ из $(a, b) \notin \notin \theta_1$ следует $(a, b) \notin \theta_2$, то $\theta_2 \subseteq \theta_1$.

Далее рассматривается конечный детерминированный вполне определенный автомат-расповит, Q – множество состояний, $\delta: Q \times X \to Q$ – функция переходов, а $F \subseteq Q$ – множество заключительных состояний. Ниже понадобится понятие обратной функции переходов $\delta^{-1}: Q \times X \to 2^Q$, определяемой как $\delta^{-1}(q,x) = \{q' \mid \delta(q', x) = q\}$. Для множества $B \subseteq Q$ обозначим $\delta(B, x) = \{\delta(q, x) \mid q \in B\}$ и $\delta^{-1}(B, x) = \{q' \in Q \mid \delta(q', x) \in B\}$.

Пусть X^* – множество всех слов в алфавите X, а $\varepsilon \in X^*$ – пустое слово. Определим расширенную функцию переходов $\delta^*: Q \times X^* \to Q$ следующим образом: $\delta^*(q, \varepsilon) = q$, $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$, где $w \in X^*$, $x \in X$. Каждому состоянию $q \in Q$ ставится в соответствие множество слов $L(q) = \{w \in X^* \mid \delta(q, w) \in F\}$, распознаваемых автоматом в этом состоянии. Состояния q_1 и q_2 называются *эквивалентными* (обозначается $q_1 \sim q_2$), тогда и только тогда, когда $L(q_1) = L(q_2)$.

Любое отношение эквивалентности θ на множестве состояний автомата A называется *конгруэнцией*, если для всех $q, q' \in Q$ и $x \in X$ из $q \theta q'$ следует $\delta(q, x)\, \theta\, \delta(q', x)$ и множество F является объединением некоторых классов эквивалентности θ. Несложно показать, что определенная ранее эквивалентность $\sim$ является максимальной (относительно включения) конгруэнцией на Q, которую обозначим θ_Q.

Утверждение 2. Разбиение $\pi = \{B_1, B_2, ..., B_m\}$ множества Q соответствует конгруэнции тогда и только тогда, когда $\forall B_i \in \pi \forall x \in X \ \exists B_j \in \pi$ $(\delta(B_i, x) \subseteq B_j)$.

Автомат A называется *минимальным*, если для любой пары (q, q') его различных состояний $L(q) \ne L(q')$, другими словами, это автомат, не имеющий эквивалентных состояний. Результат минимизации автомата A – автомат, состояния которого представляют собой классы эквивалентности конгруэнции θ_Q.

Алгоритм Хопкрофта

Минимизация детерминированного вполне определенного автомата $A = \langle X, Q, \delta, F\rangle$ состоит в нахождении максимальной конгруэнции θ_Q на множестве его состояний. Построение требуемой конгруэнции начинается с разбиения $\pi_0 = \{F, Q \backslash F\}$, которое затем последовательно уточняется путем расщепления некоторых классов до получения разбиения, соответствующего конгруэнции.

Условием расщепления класса B разбиения π является наличие таких $C \in \pi$ и $x \in X$, что $\delta(B, x) \cap C \ne \varnothing$ и $\delta(B, x) \not\subset C$. Такую пару (C, x) будем называть *сплитером* класса B. Для выражения $\delta^{-1}(C, x)$ введем сокращение $x^{-1}C$. Расщепление класса $B \in \pi$ на два класса $B' = B \cap \cap x^{-1}C = \{q \in B \mid \delta(q, x) \in C\}$ и $B'' = B \setminus x^{-1}C = \{q \in B \mid \delta(q, x) \notin C\}$ назовем *расщеплением по сплитеру* (C, x).

Используя введенные понятия, можно сформулировать понятие конгруэнции на множестве Q. Разбиение $\pi = \{B_1, B_2, ..., B_m\}$ множества Q соответствует конгруэнции, если ни один класс $B_i \in \pi$ не удовлетворяет условию расщепления, т.е. ни для каких $B_i, B_j \in \pi$ и $x \in \in X$ (B_i, x) не является сплитером класса B_j (не расщепляет B_j).

Расщепление всех классов разбиения π, расщерляемых по сплитеру (C, x), назовем *шагом расщеплений*.

Процесс минимизации представляет собой построение последовательности разбиений π_0, π_1, ..., π_k, в которой каждое разбиение π_{i+1} $(i \geq 0)$ получается из предыдущего в результате выбора блока $C \in \pi_i$ и последовательного выполнения шагов расщеплений по сплитерам (C, x) для всех $x \in X$. Для каждого класса B разбиения, полученного в результате выполнения шага расщепления по (C, x), либо $\forall q \in B\ \delta(q, x) \in C$, либо $\forall q \in B\ \delta(q, x) \notin C$. Отсюда следует, что дальнейшие расщепления по (C, x) выполнять не требуется.

Пусть $C = C_1 \cup C_2$ и $C_1 \cap C_2 = \varnothing$. Обозначим π_C разбиение, полученное из π в результате расщепления его классов по сплитеру (C, x). Аналогично обозначим π_{C_1} и π_{C_2} разбиения, полученные путем расщепления классов π соответственно по сплитерам (C_1, x) и (C_2, x).

***Лемма* 1**. Для любого $x \in X$ справедливо $\pi_C \wedge \pi_{C_1} = \pi_C \wedge \pi_{C_2} = \pi_{C_1} \wedge \pi_{C_2}$.

Доказательство. Пусть класс $B \in \pi$ расщепляется по сплитеру (C, x) на $B' = B \cap x^{-1}C$ и $B'' = B \setminus B'$, по сплитеру (C_1, x) – на $B_1' = B \cap \cap x^{-1}C_1$ и $B_1'' = B \setminus B_1'$, а по сплитеру (C_2, x) – на $B_2' = B \cap x^{-1}C_2$ и $B_2'' = B \setminus B_2'$. Фигурирующие здесь классы, на которые расщепляется B, удовлетворяют следующим соотношениям: $B' = B_1' \cup B_2'$, $B_1' \cap B_2' = \varnothing$, $B_1'' = B'' \cup B' \setminus B_1' = B'' \cup B_2'$, $B_2'' = B'' \cup B' \setminus B_2' = B'' \cup B_1'$.

В разбиении $\pi_C \wedge \pi_{C_1}$ класс B разбивается на $(B', B'') \wedge (B_1', B_1'') = \{(B' \cap B_1'), (B' \cap B_1''), (B'' \cap B_1'), (B'' \cap B_1'')\} = \{B_1', B_2', \varnothing, B''\}$.

В разбиении $\pi_C \wedge \pi_{C_2}$ класс B разбивается на $(B', B'') \wedge (B_2', B_2'') = \{(B' \cap B_2'), (B' \cap B_2''), (B'' \cap B_2'), (B'' \cap B_2'')\} = \{B_2', B_1', \varnothing, B''\}$.

Аналогично для $\pi_{C_1} \wedge \pi_{C_2}$ имеем $(B_1', B_1'') \wedge \wedge (B_2', B_2'') = \{(B_1' \cap B_2'), (B_1' \cap B_2''), (B_1'' \cap B_2'), (B_1'' \cap B_2'')\} = \{\varnothing, B_1', B_2', B''\}$.

Таким образом, в каждом из разбиений $\pi_C \wedge \wedge \pi_{C_1}$, $\pi_C \wedge \pi_{C_2}$, $\pi_{C_1} \wedge \pi_{C_2}$ класс B разбивается на одни и те же классы. Если по какому-либо из сплитеров (C_1, x) или (C_2, x) класс B не расщепляется, то все три произведения разбиений совпадают с разбиением π_C. Поскольку классы разбиения π расщепляются независимо, отсюда вытекает справедливость леммы 1.

На этой лемме основывается алгоритм Хопкрофта, поэтому она существенно используется в доказательстве его корректности.

Последовательность шагов расщеплений по сплитерам (C, x) для всех $x \in X$ будем называть *циклом расщеплений* по классу C. Поскольку нет необходимости выполнять расщепления по всем классам, порождаемым в процессе уточнения разбиений, то параллельно с уточнением текущих разбиений строится и изменяется множество L классов, по которым следует осуществлять расщепления.

Алгоритм минимизации автомата $A = \langle X, Q, \delta, F \rangle$.

```
P := {F, Q\F};
if |F| ≤ |Q\F| then
L := {F}
else
L := {Q/F};
while L ≠ ∅ do
  выбрать в L класс (скажем C) и удалить его из L
  for каждого x ∈ X do
    for каждого класса B ∈ P, для которого B ∩ x^{-1}C ≠ ∅ и B \ x^{-1}C
≠ ∅ do
      заменить B в P двумя классами B′ = B ∩ x^{-1}C и B″ = B \ x^{-1}C
       if B ∈ L then
        заменить B в L классами B′ и B″
       else
          if |B′| ≤ |B″| then
              добавить B′ в L
            else
              добавить B″ в L
    end for;
  end for;
end while;
```

В процессе работы алгоритма итеративно преобразуются два множества: текущее разбиение $\pi_i = \{B_1, ..., B_m\}$ и текущее множество L_i тех классов разбиения π_i, по которым следует выполнять расщепления.

В каждом цикле ***while*** $L \neq \varnothing$ выполняются следующие операции. Из L_i выбирается и удаляется какой-либо из классов C, затем в каждом шаге расщеплений по сплитеру (C, x), $x \in X$, каждый класс B текущего разбиения π_i, который расщепляется на B' и B'', заменяется парой классов B' и B'' и, если $B \in L_i$, он заменяется в L_i на B' и B'', в противном случае в L_i добавляется меньший по мощности класс из B' и B''. Поскольку в каждом цикле ***while*** из L удаляется один класс, количество которых конечно, то алгоритм заканчивает работу, когда $L_i = \varnothing$.

Пример 1. Автомат A задан следующей таблицей переходов.

Таблица 1

	0	1	2	3	4	5	**6**	**7**	8	**9**
a	1	5	1	4	5	5	2	8	4	5
b	3	5	5	7	3	2	9	8	9	6

Здесь $X = \{a, b\}$, $F = \{6, 7, 9\}$. Множество состояний $\{q_1, q_2, ..., q_k\}$ будем записывать в виде $(q_1q_2...q_k)$. Исходное разбиение $\pi_0 = \{B_1 = (0123458), B_2 = (679)\}$, $L_0 = \{(679)\}$.

Первый цикл: $C = (679)$.

$x = a.\ a^{-1}(679) = \varnothing$.

$x = b.\ b^{-1}(679) = (3689)$. Расщепляются оба класса.

$(0123458) \Rightarrow \{(01245), (38)\}$, $(679) \Rightarrow \{(69), (7)\}$.

$\pi_1 = \{(01245), (38), (69), (7)\}$, $L_1 = \{(38), (7)\}$.

Второй цикл: $C = (7)$.

$$x = a.\ a^{-1}(7) = \varnothing.$$

$x = b.\ b^{-1}(7) = (3)$. Расщепляется класс $(38) \Rightarrow \Rightarrow \{(3), (8)\}$.

$\pi_2 = \{(01245), (3), (8), (69), (7)\}$, $L_2 = \{(3), (8)\}$.

Третий цикл: $C = (3)$.

$$x = a.\ a^{-1}(3) = \varnothing.$$

$x = b.\ b^{-1}(3) = (04)$. Расщепляется класс $(01245) \Rightarrow \{(125), (04)\}$.

$\pi_3 = \{(125), (04), (3), (8), (69), (7)\}$, $L_3 = \{(8), (04)\}$.

Четвертый цикл: $C = (04)$.

$$x = a.\ a^{-1}(04) = (38).$$

$x = b.\ b^{-1}(04) = \varnothing$. Расщепления не происходят.

$\pi_4 = \{(125), (04), (3), (8), (69), (7)\}$, $L_4 = \{(8)\}$.

Пятый цикл: $C = (8)$.

$$x = a.\ a^{-1}(8) = (7).$$

$x = b.\ b^{-1}(8) = (7)$. Расщепления не происходят.

$\pi_5 = \{(125), (04), (3), (8), (69), (7)\}$, $L_5 = \varnothing$.

На этом работа алгоритма заканчивается. Получаем автомат с шестью состояниями, заключительным состояниям которого соответствуют классы (69) и (7).

Доказательство корректности алгоритма

Следует доказать, что алгоритм Хопкрофта решает поставленную задачу, т.е., что

- получаемое им разбиение соответствует конгруэнции относительно функции переходов;
- эта конгруэнция максимальна.

Для рассматриваемого варианта алгоритма несколько изменим понятие сплитера. Если существует такой $x \in X$, что (C, x) расщепляет класс B, то класс C будем называть сплитером класса B. Так, чтобы показать, что полученное алгоритмом разбиение π соответствует конгруэнции, достаточно показать, что ни один класс разбиения π не является сплитером ни для какого класса этого разбиения. Для этой цели удобно ввести понятие бинарного *дерева расщеплений*, представляющего последовательность всех расщеплений, выполненных в процессе работы алгоритма.

Вершинами этого дерева являются классы разбиений, порождаемых при работе алгоритма. Корень дерева соответствует множеству Q. Два его потомка – классы F и $Q\backslash F$. Потомки вершины, соответствующей классу B, представляют классы B' и B'', на которые он расщепляется при выполнении некоторого шага расщеплений. Все вершины, соответствующие классам, по которым осуществлялись расщепления, называются *закрытыми* и обозначаются черными кружками, а все остальные (открытые) – белыми. Классы разбиения, по-

лученного после окончания работы алгоритма, представлены листьями дерева расщеплений. Очевидно, что вид дерева зависит от последовательности выполненных расщеплений. Возможное дерево расщеплений для примера 1 приведено на рис. 1.

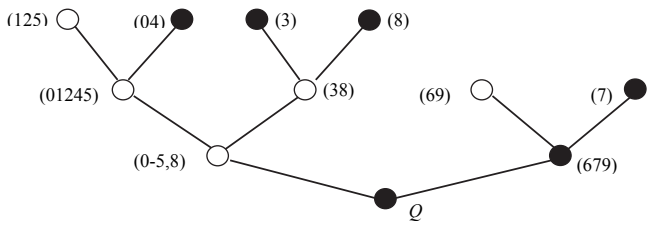

Рис. 1

Введем правила распространения закрытых вершин, позволяющих все вершины дерева считать закрытыми.

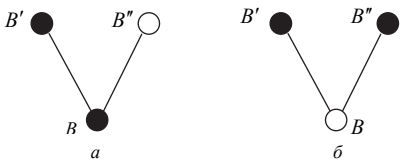

Рис. 2

Рассмотрим два фрагмента дерева расщеплений, приведенные на рис. 2. На рис. 2, *а* показано, что по сплитерам B и B' выполнялись расщепления. Из леммы 1 следует, что цикл расщеплений по B'' не изменяет полученного разбиения, поэтому вершину B'' также можно считать закрытой. Аналогичная ситуация приведена на рис. 2, *б*. Поскольку расщепления выполнялись по B' и B'', то цикл расщеплений по B полученного разбиения не приведет к его изменению. Таким образом, вершину B также можно считать закрытой.

Покажем, что, выполнив распространение закрытых вершин сначала в соответствии с правилом 2, *б*, а затем согласно правилу 2, *а*, получим дерево, все вершины которого будут закрытыми. Если все вершины-листья закрыты, то ни один из классов полученного разбиения не может служить сплитером ни для какого класса этого разбиения, из чего следует, что полученное разбиение соответствует конгруэнции на множестве Q.

Для доказательства этого утверждения рассмотрим процесс построения дерева расщеплений в соответствии с последовательностью шагов расщеплений, начиная с корня Q. На каждом шаге расщеплений к построенной части дерева добавляются вершины, полученные на этом шаге в результате расщеплений классов. Вершины, добавляемые на каждом шаге расщеплений, назовем *первичными*, если соответствующие им классы добавляются во множество L, и *вторичными* в противном случае. После окончания процесса построения дерева расщеплений, все первичные вершины, по которым осуществлялись расщепления, будут закрытыми. Первичная вершина не будет закрытой только в том случае, если она была расщеплена до того как ее выбрали в качестве сплитера. Очевидно, что оба ее потомка – также первичные вершины.

Утверждение 3. В построенном дереве расщеплений после всех возможных применений правила 2, *б* все первичные вершины будут закрытыми.

Доказательство. Пусть T – минимальное поддерево дерева расщеплений, порожденное *открытой* первичной вершиной B, т.е. не содержащее других открытых первичных вершин. Очевидно, что в силу минимальности T оба потомка вершины B закрыты. После применения правила 2, *б*, вершина B также становится закрытой. Применим это правило ко всем таким минимальным поддеревьям. Пусть теперь T_1 – наименьшее поддерево, порожденное открытой первичной вершиной, не содержащее других открытых первичных вершин. Потомки его корня либо изначально были закрытыми вершинами, либо стали такими после предыдущих применений правила 2, *б*. Применение этого правила делает его корневую вершину закрытой, и, таким образом, все первичные вершины поддерева T_1 будут закрытыми. Очевидно, что после применения правила 2, *б* в таком порядке, все первичные вершины дерева расщеплений станут закрытыми.

Утверждение 4. Если в дереве расщеплений все первичные вершины закрыты, то в результате всех возможных применений правила 2, *а* все его вершины станут закрытыми.

Это утверждение достаточно очевидно. Начиная применение правила 2, *а* с фрагмента, образованного вершиной Q и двумя ее потомками, и двигаясь затем в направлении роста дерева, получим все вершины в дереве расщеплений закрытыми.

Из этих утверждений следует справедливость утверждения 1.

Для дерева на рис. 1 распространение закрытых вершин выполняется в следующем порядке. Сначала делаем закрытой вершину (38) (правило 2, *б*), затем согласно правилу 2, *а* – вершины (0–5,8), (69), (01245) и (125).

Итак, в процессе работы алгоритма строится последовательность разбиений, соответствующих эквивалентностям $\theta_0 \supseteq \theta_1 \supseteq ... \supseteq \theta_k = \theta_{k+1}$

Ранее показано, что θ_k – конгруэнция на Q. Остается показать, что θ_k совпадает с наибольшей конгруэнцией θ_Q. Сначала, используя утверждение 1, покажем, что $\theta_i \supseteq \theta_Q$ для всех $i \geq 0$.

Индукцией по последовательности эквивалентностей докажем, что для всех $i \geq 0$ из $(a,b) \notin \theta_i$ следует $(a, b) \notin \theta_Q$. Это очевидно для θ_0, так как пара, состоящая из заключительного и незаключительного состояний, не принадлежит θ_Q. Предположим, что это утверждение выполняется для всех $0 \leq l \leq i$. Поскольку $\theta_{i+1} \subseteq \theta_i$, то найдется такая пара $(a, b) \in \theta_i$, что $(a, b) \notin \notin \theta_{i+1}$, а значит, для некоторого $x \in X$ $(x(a), x(b)) \notin \theta_i$ (условие расщепления, приводящего к θ_{i+1}). Отсюда согласно индуктивному предположению следует, что $(x(a), x(b)) \notin \theta_Q$. Так как θ_Q – конгруэнция, то в силу $(x(a), x(b)) \notin \notin \theta_Q$ пара (a, b) также не принадлежит θ_Q. Таким образом, $(a, b) \notin \theta_{i+1} \Rightarrow (a, b) \notin \theta_Q$ и, следовательно, $\theta_0 \supseteq \theta_1 \supseteq ... \supseteq \theta_k \supseteq \theta_Q$. Поскольку θ_Q – наибольшая конгруэнция на Q, а θ_k – конгруэнция, то $\theta_k = \theta_Q$.

Анализ временной сложности алгоритма

Утверждение 5. Для автомата с n состояниями и k-буквенным алфавитом алгоритм Хопкрофта может быть реализован в худшем случае с временной сложностью $O(kn \log n)$.

Для получения этой оценки процесс построения конгруэнции θ_Q должен быть реализован так, чтобы используемые в нем основные операции имели следующие временные характеристики:

• доступ к классу, которому принадлежит заданное состояние, осуществляется за время, определяемое константой;

• просмотр всех элементов класса осуществляется за время, пропорциональное его размеру (мощности);

• добавление или удаление элемента множества осуществляется за время, определяемое константой.

Существует несколько реализаций структур данных, удовлетворяющих указанным временным ограничениям. Такие структуры данных описаны в [3, 5, 6].

Прежде всего покажем, что при выполнении указанных требований один шаг расщеплений по сплитеру (C, x) может быть реализован за время $O(|(x^{-1}C)|)$. Ниже приведен возможный вариант такой реализации.

• Для каждого состояния $q \in x^{-1}C$ класс B, содержащий q, отмечается как кандидат на расщепление, состояние q добавляется в список состояний, удаляемых из B, и счетчик количества состояний в этом списке увеличивается на единицу.

• Отмеченный класс расщепляется, если количество состояний в списке отличается от размера класса. При этом состояния, имеющиеся в списке, удаляются из класса B и образуют новый класс B''.

Множество L реализуется так, что наличие в нем рассматриваемого сплитера проверяется за константное время, и сплитер добавляется и удаляется также за константное время. Это позволяет заменить сплитер B сплитерами B' и B'' за это же время, поскольку B' представляет собой модифицированный процедурой расщепления класс B, и достаточно добавить только сплитер B''.

С учетом всех замечаний приведем основную часть доказательства утверждения 5. Класс из L, содержащий состояние q, назовем q-сплитером. Для некоторого выполнения алгоритма рассмотрим последовательность q-сплитеров, по которым осуществлялось расщепление. Каждый новый такой q-сплитер добавляется в L

только после удаления из L предшествовавшего q-сплитера в начале выполнения цикла расщеплений по нему. Если расщепление класса происходит до его удаления из L, то этот класс не используется в качестве сплитера и в рассматриваемую последовательность не входит. Очевидно, что каждый последующий член рассматриваемой последовательности по мощности не менее чем в два раза меньше предыдущего. Таким образом, количество q-сплитеров, по которым осуществлялось расщепление, не превышает $\log n$.

Утверждение 6. Суммарное количество состояний во всех множествах вида $x^{-1}C$, где (C, x) – сплитер, по которому осуществлялось расщепление, не превышает $kn \log n$.

Доказательство. Рассмотрим какой-либо переход, скажем, из состояния q' в состояние q, соответствующий $\delta(q', x) = q$. Если (C, x) – q-сплитер, то состояние q' принадлежит $x^{-1}C$. Очевидно, что множествам $x^{-1}C$, где (C, x) не является q-сплитером, оно не принадлежит. Ранее было показано, что количество q-сплитеров, по которым проводилось расщепление, не превышает $\log n$. Таким образом, состояние q' может встречаться во множествах $x^{-1}C$, где (C, x) – q-сплитеры, по которым проводилось расщепление, не более $\log n$ раз. Поскольку в автомате имеется ровно kn переходов, то сумма мощностей всех множеств $x^{-1}C$ не превосходит $kn \log n$. Заметим, что не для каждого перехода в состояние q имеется q-сплитер, по которому проводилось расщепление.

Итак, суммарное время выполнения всех шагов расщепления ограничено сверху величиной $O(kn \log n)$. Исходя из аналогичных рассуждений, можно показать, что суммарное время, затрачиваемое на построение всех множеств вида $x^{-1}C$, имеет такую же верхнюю оценку. Временная сложность каждой из остальных операций, время выполнения которых не определяется константой, оценивается как $O(kn)$. Следовательно, время работы всего алгоритма оценивается сверху величиной $O(kn \log n)$.

Выполнение алгоритма недетерминировано, поскольку выбор очередного сплитера из множества L может быть произвольным. Различные стратегии этого выбора могут давать различное время работы алгоритма, однако не изменяют оценку его временной сложности. В работе [8] показано, что эта оценка достижима для некоторого класса автоматов при определенной стратегии выбора элемента из множества L.

Реализация структур данных

Далее описаны структуры данных [6], удовлетворяющие требованиям, необходимым для получения приведенной ранее оценки временной сложности алгоритма.

Имена состояний и классов разбиений представляются числами 0, 1, 2, ... Подмножества множества Q представляются как сегменты массивов фиксированной длины.

Отношение $x^{-1}(q)$ задается двумя матрицами: ***inv_head*** и ***inv_elts***.

inv_elts – $(|X|\times|Q|)$-матрица элементов сегментов, соответствующих значениям $x^{-1}(q)$. В строках этой матрицы состояния перечислены в том порядке, как они встречаются в таблице $x^{-1}(q)$.

inv_head – $(|X|\times|Q|)$-матрица заголовков сегментов. Заголовок сегмента – это пара $\langle i, k\rangle$, где i – индекс в массиве ***inv_elts*** $[x]$ первого элемента сегмента, а k – количество элементов сегмента, уменьшенное на еденицу. Этой паре соответствуют поля ***first*** и ***size*** в записи, состоящей из двух элементов.

Обратное отношение переходов $x^{-1}(q)$ для примера 1 представлено таблицей:

Таблица 2

	0	1	2	3	4	5	6	7	8	9
a	∅	(02)	(6)	∅	(38)	(1459)	∅	∅	(7)	∅
b	∅	∅	(5)	(04)	∅	(12)	(9)	(3)	(7)	(68)

Для рассматриваемого примера матрица ***inv_elts*** имеет вид

	0	1	2	3	4	5	6	7	8	9
a	0	2	6	3	8	1	4	5	9	7
b	5	0	4	1	2	9	3	7	6	8

Таблица ***inv_head***

	0	1	2	3	4	5	6	7	8	9
a	∅	⟨0,1⟩	⟨2,0⟩	∅	⟨3,1⟩	⟨5,3⟩	∅	∅	⟨9,0⟩	∅
b	∅	∅	⟨0,0⟩	⟨1,1⟩	∅	⟨3,1⟩	⟨5,0⟩	⟨6,0⟩	⟨7,0⟩	⟨8,1⟩

Например, ***inv_elts*** [a][***inv_head*** [a][8].*first*] = = 7, где ***inv_head*** [a][8].*first* = 9.

Аналогично представляются и разбиения множества Q.

cls_head – массив размера $|Q|$ заголовков классов разбиений. Индексами массива являются имена классов (0, 1, ...). Заголовок сегмента (класса) имеет четыре поля: ***first***, ***size***, ***counter*** и ***move***. В поле ***counter*** для класса B подсчитывается количество состояний в $x^{-1}C \cap$ $\cap B$, а в поле **move** строится список, представляющий множество $x^{-1}C \cap B$.

cls_elts – массив размера $|Q|$ состояний, сегменты которого соответствуют классам.

Для разбиения π_1 = {(01245), (38), (69), (7)} соответствующие массивы имеют вид

Массив ***cls_elts***

0	1	2	3	4	5	6	7	8	9
0	1	2	4	5	3	8	6	9	7

Массив ***cls_head***

0	1	2	3	4	5	6	7	8	9
⟨0,4,0⟩	⟨5,1,0⟩	⟨7,1,0⟩	⟨9,0,0⟩	∅	∅	∅	∅	∅	∅

Здесь в заголовках поле ***move*** не представлено.

Исходное разбиение π_0 = {(0123458), (679)} задается следующим образом:

Массив ***cls_elts***

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	8	6	7	9

Массив ***cls_head***

0	1	2	3	4	5	6	7	8	9
⟨0,6,0⟩	⟨7,2,0⟩	∅	∅	∅	∅	∅	∅	∅	∅

Каждому состоянию $q \in Q$ соответствует класс, которому оно принадлежит. Это соответствие задается массивом ***states***, каждый элемент ***states***[q] которого содержит следующие поля:

cls_of – номер соответствующего класса и

idx_of – индекс состояния q в соответствующем сегменте массива ***cls_elts***.

Для разбиения π_1 = {(01245), (38), (69), (7)} массив ***states*** имеет вид

0	1	2	3	4	5	6	7	8	9
⟨0,0⟩	⟨0,1⟩	⟨0,2⟩	⟨1,5⟩	⟨0,3⟩	⟨0,4⟩	⟨2,7⟩	⟨3,9⟩	⟨1,6⟩	⟨2,8⟩

Классы исходного разбиения $\pi_0 = \{F,\ Q \backslash F\}$ нумеруются соответственно числами 0 и 1.

При расщеплении класса 0 получаются классы 0 и 2, а при расщеплении класса 1 – 1 и 3. Таким образом, значения номеров классов не превышают $|Q| - 1$.

Минимизация автоматов Мура и Мили

Автоматы рассматриваемых классов имеют следующий вид:

Автомат Мура $A = \langle X, Q, Y, \delta, \mu \rangle$, где X, Q, Y – соответственно входной алфавит, множество состояний и выходной алфавит, $\delta: Q \times X \to$ $\to Q$ – функция переходов, μ – функция выходов (отметок состояний).

Автомат Мили $A = \langle X, Q, Y, \delta, \lambda \rangle$, где X, Q, Y, δ – то же, что и выше, а функция выходов λ имеет вид λ: $Q \times X \to Y$.

Поскольку задача минимизации автомата основывается на понятии эквивалентности состояний, приведем соответствующие определения для рассматриваемых автоматов.

Для автомата Мура $q_1 \sim q_2$, если $\forall w \in X^*$ $\mu(\delta^*(q_1, w)) = \mu(\delta^*(q_2, w))$.

Для автомата Мили $q_1 \sim q_2$, если $\forall w \in X^*$ $\lambda^*(q_1, w)) = \lambda^*(q_2, w))$. Здесь $\lambda^*(q, x_1 x_2 \ldots x_m)) =$ $= \lambda(\delta^*(q, x_1 x_2 \ldots x_{m-1}), x_m)$, где $x_1 x_2 \ldots x_m \in X^*$.

Для того чтобы по эквивалентности $\sim$ на множестве Q можно было построить минимальный автомат, она должна быть конгруэнцией. Соответствующие конгруэнции для автоматов Мура и Мили определяются следующими требованиями.

Для автомата Мура из $q_1 \sim q_2$ следует $\forall x \in X(\delta(q_1, x) \sim \delta(q_2, x))$ и $\mu(q_1) = \mu(q_2)$, т.е. все состояния в каждом классе эквивалентности должны быть отмечены одним и тем же выходным сигналом. Для автомата Мили из $q_1 \sim q_2$ следует $\forall x \in X(\delta(q_1, x) \sim \delta(q_2, x))$ и $\forall x \in X(\lambda(q_1, x) = \lambda(q_2, x))$. Пусть множество X линейно упорядочено, т.е. $X = \langle x_1, \ldots, x_k \rangle$. Для каждого $q \in Q$ рассмотрим вектор $\langle \lambda(q, x_1), \ldots$

$\ldots, \lambda(q, x_k)\rangle$, который обозначим $\overline{\lambda}(q)$, тогда вторую часть условия конгруэнтности для автомата Мили можно записать так: из $q_1 \sim q_2$ следует $\overline{\lambda}(q_1) = \overline{\lambda}(q_2)$, т.е. для всех состояний, принадлежащих одному и тому же классу конгруэнции, векторы $\overline{\lambda}(q)$ должны совпадать. Таким образом, каждый класс начального разбиения π_0 в алгоритме Хопкрофта для автомата Мура определяется значением выходного сигнала на его состояниях, а для автомата Мили – значением вектора $\overline{\lambda}(q)$. Итак, для автомата-распознавателя $|\pi_0| = 2$, для автомата Мура $|\pi_0| = k$, для автомата Мили $|\pi_0| \le |Y|^k$. С этим же связано и различие в начальных значениях множества L, которое состоит из всех, кроме одного, элементов начального разбиения. Остальная часть алгоритма для всех случаев совпадает. Если время построения начального разбиения множества оценивается как O(kn), то оценка временной сложности алгоритма для всех рассмотренных видов автоматов будет одной и той же.

Пример 2. Автомат Мили задан следующими таблицами переходов и выходов.

Таблица 3

	1	2	3	4	5	6	7	8
x	1	2	1	2	1	1	2	1
y	8	8	7	2	2	3	5	4
z	4	5	7	5	4	6	3	3

Таблица 4

	1	2	3	4	5	6	7	8
x	u	u	u	v	v	v	u	u
y	v	v	v	u	u	u	u	u
z	u	u	u	v	v	v	v	v

Как следует из таблицы выходов, имеется три класса: (123), (456), (78), составляющих исходное разбиение π_0.

Построим таблицу, задающую отношение $x^{-1}(q)$, обратное отношению переходов.

Таблица 5

	1	2	3	4	5	6	7	8
x	1,3,5,6,8	2,4,7	∅	∅	∅	∅	∅	∅
y	∅	4,5	6	8	7	∅	3	1,2
z	∅	∅	7,8	1,5	2,4	6	3	∅

Итак, $\pi_0 = \{(123), (456), (78)\}$. Пусть значение L_0 равно $\{(123), (456)\}$.

Первый цикл: $C = (123)$

$x^{-1}(123) = Q$, $y^{-1}(123) = (456)$, $z^{-1}(123) = (78)$.

Ни один класс не расщепляется.

$\pi_1 = \{(123), (456), (78)\}$, $L_1 = \{(456)\}$.

Второй цикл: $C = (456)$.

$x^{-1}(456) = \varnothing$, $y^{-1}(456) = (78)$, $z^{-1}(456) = (12456)$.

Расщепляется класс $(123) \Rightarrow (12), (3)$.

$\pi_2 = \{(12), (3), (456), (78)\}$, $L_2 = \{(3)\}$

Третий цикл: $C = (3)$.

$x^{-1}(3) = \varnothing$, $y^{-1}(3) = (6)$, $z^{-1}(3) = (78)$.

Расщепляется класс $(456) \Rightarrow (45), (6)$.

$\pi_3 = \{(12), (3), (45), (6), (78)\}$. $L_3 = \{(6)\}$.

Четвертый цикл: $C = (6)$.

$x^{-1}(6) = \varnothing$, $y^{-1}(6) = (\varnothing)$, $z^{-1}(6) = (6)$.

Ни один класс не расщепляется.

$\pi_4 = \{(12), (3), (45), (6), (78)\}$, $L_4 = \varnothing$.

На этом работа алгоритма заканчивается. После перенумерации состояний получаем автомат, задаваемый следующими таблицами переходов и выходов.

Таблица 6

	1	2	3	4	5
x	1	1	1	1	1
y	5	5	1	2	3
z	3	5	3	4	2

Таблица 7

	1	2	3	4	5
x	u	u	v	v	u
y	v	v	u	u	u
z	u	u	v	v	v

Заключение. Работа посвящена описанию и анализу алгоритма минимизации конечных детерминированных вполне определенных автоматов, предложенного Дж. Хопкрофтом в 1971 г. Из-за того, что в оригинальной работе Хопкрофта дано довольно громоздкое и недостаточно строгое описание как самого алгоритма, так и его обоснования, появился ряд последующих работ, в которых уточнялся алгоритм и давалось более строгое обоснование его корректности. Учитывая практическое отсутствие рус-

скоязычных работ такого рода, в настоящей статье сделана еще одна попытка дать довольно простое обоснование как корректности алгоритма, так и оценки его временной сложности.

Рассмотрен один из вариантов алгоритма, в котором в качестве элементов списка сплитеров, по которым следует проводить расщепления, используются классы разбиений, а не пары (класс, входной символ), как в оригинальном алгоритме и в ряде последующих публикаций. Приведено достаточно простое доказательство основной леммы (лемма 1), на которой основано получение временной сложности $O(kn \log n)$. Из этой же леммы следует, что начальное значение L_0 множества L состоит из одного класса, если множество Q рассматривать как его тривиальное разбиение. Для обоснования одного из условий корректности алгоритма предложено использовать бинарное дерево расщеплений и метод распространения закрытых состояний. Поскольку полученная Хопкрофтом оценка временной сложности налагает некоторые ограничения на реализацию алгоритма, в настоящей статье приведена формулировка этих ограничений и возможный вариант подходящей реализации используемых структур данных.

1. *Huffman D.A.* The synthesis of sequential switching circuits // J. Franklin Inst. – 1954. – **257**, N 3. – P. 161–190; N 4. – P. 275–303.
2. *Moore E.F.* Gedanken-experiments on sequential Machines // Automata studies, Princeton Univ. Press, Princeton, N.J., 1956. – P. 129–153.
3. *Hopcroft J.* An n log n algorithm for minimizing states in a finite automaton / Z. Kohavi, A. Paz (Eds.) // Proc. Internat. Symp. on the Theory of Machines and Computations, Academic Press, New York, 1971. – P. 189–196.
4. *Gries D.* Describing an algorithm by Hopcroft // Acta Inform. – 1973. – N 2. – P. 97–109.
5. *Брауэр В.* Введение в теорию конечных автоматов. – М.: Радио и связь, 1987. – 392 с.
6. *Knuutila T.* Re-describing an algorithm by Hopcroft. // Theoret. Comput. Sci. – 2001. – **250**. – P. – 333–363.
7. *Baclet M.*, *Pagetti C.* Around Hopcroft's algorithm // Lecture Notes in Comp. Sci. – 2006. – **4094**, Springer-Verlag. – P. 114–125.
8. *Berstel J.*, *Carton O*. On the complexity of Hopcroft's state minimization algorithm // Ibid. – 2005 / – **3317**, Springer-Verlag. – P. 35–44.

Поступила 11.03.2015

Тел. для справок: +38 044 525-3677; 526-3204 (Киев)
E-mail:*ancheb.@gmail.com*

UDC 519.713

A.N. Chebotarev

On Automata Minimization by Hopcroft's Algorithm

Keywords: Hopcroft's algorithm, automaton minimization, partition, congruence, time complexity, splitting tree.

The algorithm for minimizing deterministic complete finite state automata proposed by J. Hopcroft is considered. Although the existence of this algorithm is widely known, its theoretical justification is not that obvious. The aim of this study is to give a clear and understandable presentation of the algorithm focusing on the firm theoretical basis, rigorous correctness proof and time complexity analysis.

The algorithm is based on the partition notion of the automaton states which defines the equivalence relation. It constructs a sequence of such partitions by a stepwise refinement of some initial partition. The last partition in this sequence corresponds to the maximal congruence on the automaton's states, i.e., the equivalence stable with respect to the transition function of the automaton. To prove correctness of the algorithm it is necessary to show that the final constructed partition is a congruence, and that this congruence is maximal. In the first part of this proof, the introduced notion of binary splitting tree is used whose nodes are classes of the partitions constructing by the algorithm and a technique for extending the so called "closed" nodes of this tree. The second part is based on the proposition concerning the order on partitions. It is proved that every partition in the sequence generated by the algorithm is greater than or equal to the partition corresponding to the maximal congruence.

It is show that Hopcroft's algorithm can be implemented to worst-case time complexity $O(kn \log n)$ for an automaton with n states over a k-letter alphabet. This statement relies on the fact that the total number of actions refining the partitions over all steps of algorithm execution is of the order of $O(kn \log n)$. The data structures which allow obtaining this time complexity is presented. The last section describes the algorithm features for minimizing Moore and Mealy automata.

●