УДК 510.5

В. И. Поляков, В. И. Скорубский

ПРЕОБРАЗОВАНИЕ МОДЕЛЕЙ АЛГОРИТМОВ

Рассматриваются преобразования регулярных выражений в конечные автоматы и обратные преобразования, относящиеся к теории алгоритмов и автоматов. Приведены преобразования этих моделей в блок-схемы и обратно.

Ключевые слова: *регулярные языки, регулярные выражения, конечные автоматы, модели алгоритмов, блок-схемы.*

Введение. Практика программирования предполагает решение задачи, а не получение доказательства того, что она разрешима. Вместе с тем в теории алгоритмов исследованы методы и модели, которые используются в программировании. К числу таких моделей относят регулярные языки и конечные автоматы. В практической реализации алгоритмов используются алгоритмические языки и блок-схемы. В настоящей работе рассматриваются преобразования моделей в блок-схемы и обратно.

Представить формальное описание алгоритма можно с помощью формального описания решаемой задачи, в то время как имеется неформальное (вербальное) описание задачи и соответственно переход к алгоритму будет также неформальным, и требуются верификация, тестирование и многократные итерации для приближения к допустимому решению.

Формализованное описание может быть получено с помощью регулярных языков [1, 2]. Ставится задача разработки алгоритма распознавания принадлежности фрагмента любого текста конкретному регулярному языку; доказывается, что регулярный язык может быть формально преобразован в *модель алгоритма* решения этой задачи за конечное число шагов. Такой моделью является *конечный автомат* (КА).

Расширения регулярных языков находят применение при описании лексики формальных алгоритмических языков, при описании алгоритмов редактирования, поиска по шаблону, в современном программировании (языки Perl, Java, Phyton, C#, PHP), в компиляторах и системах управления обработкой данных, реализованных в операционных системах. Следовательно, для алгоритмического решения таких задач может быть использован конечный автомат.

Преобразование регулярных выражений в конечные автоматы. Для любого регулярного языка, представленного регулярным выражением (РВ), можно построить КА — распознаватель слов, допустимых в языке.

Рассмотренная в работе [2] методика преобразования может быть упрощена с учетом алгебраических свойств операций регулярного выражения:

— последовательности символов, помещенных в скобки с операцией сложения, имеют общее начальное состояние перед открывающей скобкой и конечное после закрывающей;

— цикл может иметь произвольное число повторений. При пустом числе циклов (итераций) сохраняется начальное состояние;

— в конкатенации нескольких символов различимые состояния заменяют конкатенацией пары символов. Например, в следующем регулярном выражении определены состояния в правильных арифметических операторах и выбраны позиции — состояния КА [1]:

$$L(M)=(ab + b)((c + a)b)^*,$$
$$0\ \ 12\ 2\ \ 2\qquad 32$$

где $L(M)$ — язык, допускаемый автоматом M; a, b, c — символы входного языка КА; $((c + a)b)^*$ — множество всех цепочек, начинающихся с символов c или a и оканчивающихся символом b; 0 — начальное, 2 — конечное, 1 и 3 — промежуточные состояния.

Общие оценки числа состояний и переходов КА определяются следующим образом:

— число переходов (ребер графа) КА равно n+1, где n — число букв в регулярном выражении (в данном примере n=6);

— в полностью определенном КА нижнюю границу числа состояний m определяют из условия ms=n+1, где s — число символов входного алфавита (в данном случае s=3, $3m$=7 и m=]7/3[=3);

— если ms>n+1, то автомат является частично определенным и верхняя граница числа состояний $k \le n$+1 — количество позиций в регулярном выражении (в данном примере k=7).

Состояния размещаются в выбранные для них позиции и формируются тройки (q_i w q_j), обозначающие переходы из состояния q_i в q_j, w — символ входной цепочки символов предложения регулярного языка:

$$L(M)=\underset{0}{(}\underset{1}{ab}+\underset{2}{b}\underset{2}{)}((c+a)b)^* = ((0a1)(1b2)+(0b2))\,((2(c+a)3)(3b2))^* =$$

$$=0(a1b+b)2((c+a)3b2)^*.$$

Линейная помеченная строка символов заменяется графом КА. На рис. 1 приведен детерминированный КА, распознающий символы языка $L(M)$. Начальное состояние 0 указано направленной на него стрелкой, а конечное 2 обведено жирным кружком.

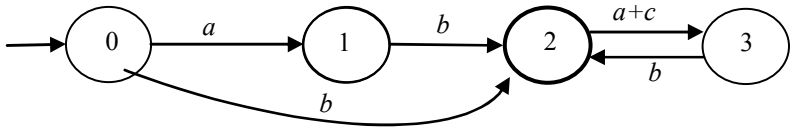

Рис. 1

Состояния q_i и q_j связаны дугой (q_i, q_j), если существует смежный символ в отмеченном регулярном выражении, и дуга помечается этим символом.

Преобразование недетерминированного конечного автомата в детерминированный. В общем случае регулярное выражение может быть преобразовано в недетерминированный конечный автомат (НДКА).

Пример. Рассмотрим следующее регулярное выражение:

$$L(M)=\underset{0}{a}\underset{1}{a}^*\underset{2}{b}d^* + ad^* = (aa^*b + a)d^* =(0a1(a1)^*b + 0a)2(d2)^*.$$

Недетерминированный КА, распознающий входные цепочки символов языка $L(M)$, представлен на рис. 2.

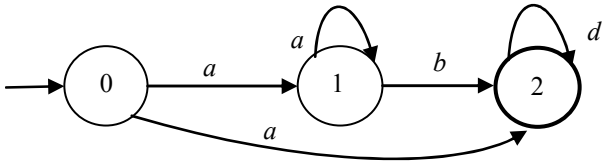

Рис. 2

Утверждение. Для НДКА можно построить эквивалентный ДКА.

Преобразование по методике, описанной в работе [2], можно представить следующей упрощенной процедурой:

1) на i-м шаге множество состояний ДКА обозначим Q_i. Q_0 обозначает множество состояний в НДКА;

2) находим различимые подмножества следующих состояний для всех условий $\sum$, применяемых к Q_i, и включаем их в множество Q_{i+1}. Итерации повторяются, пока $Q_i \ne Q_{i+1}$;

3) на каждом шаге недетерминированные переходы в состояния $\{q_i\}$ и $\{q_j\}$ становятся детерминированными при объединении их в одно состояние $\{\{1\},\{2\}\}$. Детерминированные выходы из $\{\{1\},\{2\}\}$ сохраняются.

Лемма. Число состояний в ДКА не превышает 2^m–1, где m — число состояний в НДКА.

Число 2^m–1 — количество собственных подмножеств множества, состоящего из m различимых элементов. Следовательно, процедура конечна.

Выполним преобразование НДКА в ДКА для автомата, представленного на рис. 2:

$$Q_0 = \{0, 1, 2\},$$
$$Q_1 = \{0, 1, 2, \{1, 2\}\}.$$

Эквивалентный детерминированный автомат содержит четыре состояния и определен тем же регулярным выражением $L(M')=L(M)$, состояния {2} и {1, 2} включают финальное состояние {2} из НДКА и становятся финальными в ДКА. Недетерминированные переходы в состояния {1} и {2} становятся детерминированными при объединении их в {1, 2}. Детерминированные выходы из {1, 2} сохраняются (рис. 3).

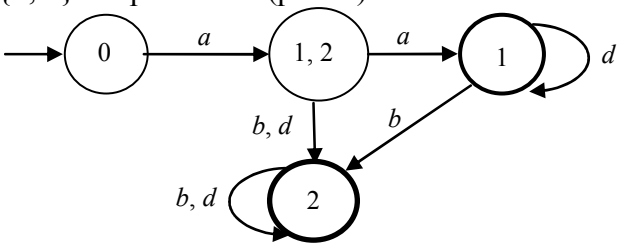

Рис. 3

Преобразование детерминированного конечного автомата в регулярное выражение может быть полезно, если КА является естественной формой описания алгоритма в задачах логического управления или получен преобразованием блок-схем.

Конечные автоматы могут применяться в качестве моделей алгоритмов для непосредственного программирования задач управления объектами [3].

К регулярному выражению применяются алгебраические преобразования и повторное обратное преобразование, позволяющее получить другие эквивалентные модели алгоритма (НДКА, минимальные ДКА и др.).

Пример. Дан КА (рис. 4).

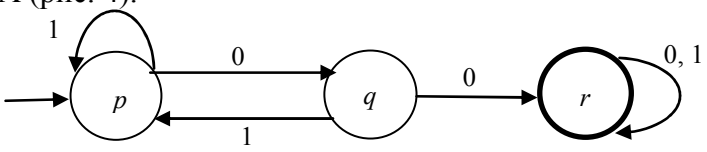

Рис. 4

Рассмотрим метод прямого преобразования КА в регулярное выражение:

1) выбрать циклы и для них последовательно представить дуги тройками вида $(q_i\, w\, g_j)$;

2) смежные дуги заменить последовательно тройками

$$L(M) = p(p1p)^*(p0q1p)^*p0q0r(r(0+1)r)^*;$$

3) пропустить первое состояние и сохранить полученное выражение

$$L(M) = p(1p)^*(0q1p)^*p0q0r((0+1)r)^*;$$

4) отметки-состояния рассматривать как позиции в регулярном выражении и выписать их

$$L(M) = p(1p)^*(0q1p)^*p0q0r((0+1)r)^* =$$
$$p\ \ p\quad\ q\ p\ \ p\ q\ r\quad\ \ r$$
$$= (1)^*(0\ 1)^*0\ 0(0+1)^* = (1^*+0\ 1)^*0\ 0(0+1)^*.$$
$$p\ \ p\ \ q\ p\ p q\ r\quad\ \ r\quad\ \ p\quad p\ q\quad p\ q\ r$$

Преобразование блок-схемы алгоритма в конечный автомат. Используя конструктивный метод преобразования блок-схем алгоритмов в КА [4], выполним преобразование на примере блок-схемы алгоритма умножения „в столбик“ $S=A*B$.

Содержательное описание алгоритма умножения:

— множимое $A = (a_{n-1}\, a_{n-2} \ldots a_1\, a_0)$ — n-разрядное десятичное число;

— множитель $B = (b_{n-1}\, b_{n-2} \ldots b_1\, b_0)$ — n-разрядное десятичное число;

— произведение S — $2n$-разрядное десятичное число, которое вычисляется суммированием частичных произведений со сдвигом влево на один десятичный разряд:

$$S = \begin{array}{r} A \bullet b_0 \\ A \bullet b_1 \\ \ldots\ldots\ldots\ldots\ldots \\ A \bullet b_n \\ \hline S_{2n-1}\, S_{2n-2} \ldots S_0. \end{array}$$

В результате получена рекуррентная формула вычисления частичных произведений $S_{i+1} = S_i + 10\,(A \bullet b_i)$. Предполагается, что вычисление частичных произведений, сдвиг и суммирование — эффективные операции. Блок-схема алгоритма умножения приведена на рис. 5.

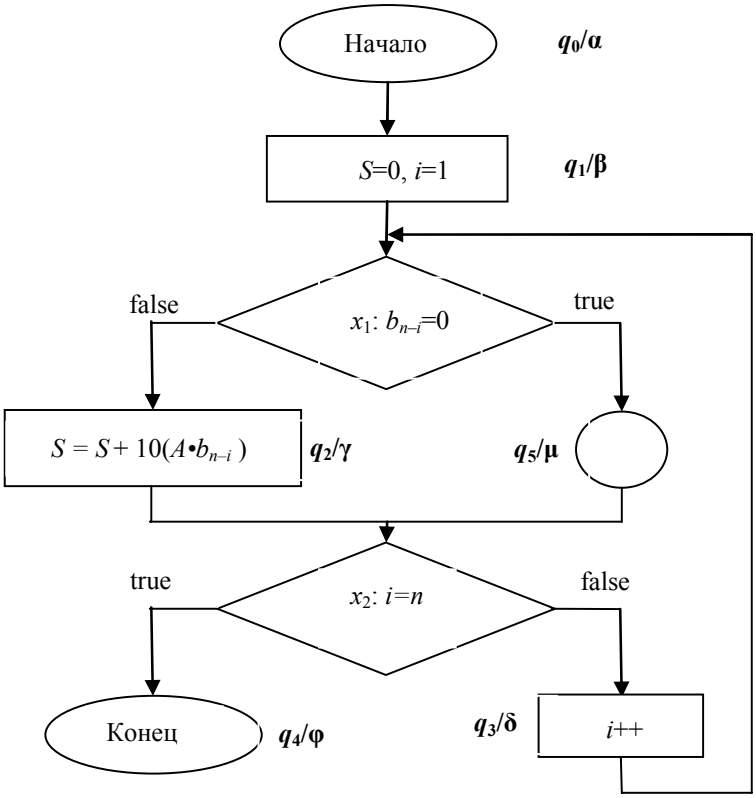

Рис. 5

Операторные вершины, а также начальная (ввод) и конечная (вывод) обозначаются как состояния КА, они обеспечивают *задержку*, необходимую для выполнения соответствующих операций. Задержки-состояния можно выбирать также для упрощения условий ветвления. В данном случае для этой цели можно включить промежуточное состояние q_5.

Таким образом, получено множество состояний $Q = \{q_0, q_1, ..., q_4, q_5\}$. Входной алфавит определяет множество символов, кодирующих обозначенные в условных вершинах предикаты и принимающие двоичные значения {true, false}. Переходы из одного состояния в другое могут быть представлены конъюнкциями значений входных переменных $\{x_1, x_2\}$.

Функции переходов КА определены в таблице.

q	q_1	q_2	q_3	q_4	q_5
q_0	T	—	—	—	—
q_1	—	$\neg x_1$	—	—	x_1
q_2	—	—	$\neg x_2$	x_2	—
q_3	—	$\neg x_1$	—	—	x_1
q_4	—	—	—	—	—
q_5	—	—	$\neg x_2$	x_2	—

Функции выходов, формируемых в состояниях, обозначим символами-командами выходного алфавита $W=\{\alpha, \beta, \delta, \gamma, \varphi, \mu\}$ и поставим им в соответствие состояния КА. Очевидно, что модель алгоритма в виде КА легко преобразуется в блок-схему, следовательно, всегда может быть выполнен переход к формальному описанию алгоритма на алгоритмическом языке и в виде программы.

При преобразованиях алгоритмов, представленных блок-схемами, могут быть использованы *частичные КА*.

Особенности использования частичных автоматов в программировании зависят от типа задачи, решаемой алгоритмическим методом:

— для распознавателей языка — ограничение на регулярный язык;

— для алгоритмов преобразования данных — ограничение на область значений данных и результатов выполнения преобразований;

— для алгоритмов управления — ограничение на входные данные.

В случае распознавателей предложений языка *при тестировании* признаки частичных автоматов могут быть использованы для контроля и выявления предложений, не принадлежащих языку.

Доопределение переходов можно использовать для контроля и тестирования программы в процессе исполнения. Для этого выполняется замена логических условий символами входного алфавита $\Sigma^*=\{T, a, b, c, d\}$, где $a= \neg x_1$, $b= x_1$, $c= \neg x_2$, $d= x_2$, q_0 — начальное и q_4 — финальное состояние:

$$L(M) =(0T1)\{[(1a2)(((2c3)(3a2))^*(2d4)+(2c3)((3b5)(5c3))^*(3b5)(5d4)] +$$

$$+(1b5)[((5c3)(3b5))^*(5d4)+(5c3)((3b5)(5c3))^*((3a2)(2c3))^*(3a2)(2d4)]\}=$$

$$=T[a((ca)^*d+c(bc)^*bd)+b((cb)^*d+c(bc)^*(ac)^*ad]=$$

$$=T[a((ca)^*+c(bc)^*b)+b((cb)^*+c(bc)^*(ac)^*a]d=$$

$$=T[a((ca)^* +c(bc)^*b) +b((cb)^* +c(bc)^*(ac)^*a]d=$$

$$= T[a((ca)^* +tb) +b((cb)^* +t(ac)^*a]d,$$

t — переводит КА в состояние, которое запускает КА, распознающий строку $c(bc)^*$.

Модель алгоритма умножения в символах входного алфавита приведена на рис. 6.

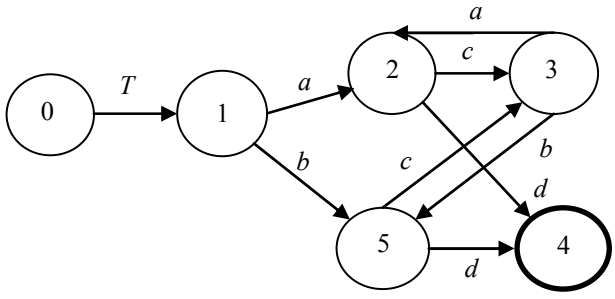

Рис. 6

Заключение. Алгоритм может быть представлен в формальной записи – конечными автоматами, синтаксическим разбором, рекурсивными функциями, регулярными выражениями. В работе рассмотрены методы преобразования моделей алгоритмов: регулярных выражений в конечные автоматы; недетерминированного конечного автомата в детерминированный; детерминированного конечного автомата в регулярное выражение; блок-схем в конечные автоматы и обратно. С помощью приведенных преобразований можно проектировать программы вычислительных машин. Скомпилированные по этим моделям программы не требуют верификации, так как получение этих моделей является доказательством существования алгоритма.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (грант № 12-07-00376-а).

СПИСОК ЛИТЕРАТУРЫ

1. *Ахо А.., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.

2. *Хопкрофт Дж., Мотвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. М.: Изд. дом „Вильямс“, 2002. 528 с.

3. *Шалыто А. А.* Switch-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998. 628 с.

4. *Майоров С. А., Новиков Г. И., Немолочнов О. Ф.* и др. Проектирование цифровых вычислительных машин. М.: Высш. школа, 1972. 344 с.

Сведения об авторах

Владимир Иванович Поляков — канд. техн. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: v_i_polyakov@mail.ru

Владимир Иванович Скорубский — канд. техн. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: vlis@km.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.