ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

Рязанов Ю.Д., магистрант
Белгородский государственный технологический университет им. В.Г. Шухова

ПРЕОБРАЗОВАНИЕ РАСПОЗНАВАТЕЛЯ С МАГАЗИННОЙ ПАМЯТЬЮ И КОНЕЧНЫМ МНОЖЕСТВОМ СОСТОЯНИЙ В РАСПОЗНАВАТЕЛЬ С ОДНИМ СОСТОЯНИЕМ

Ryazanov.iurij@yandex.ru

В статье рассматривается задача распознавания контекстно-свободных языков. Для ее решения используются распознаватели с магазинной памятью, которые могут иметь конечное множество состояний. В работе определяется класс распознавателей с магазинной памятью и конечным множеством состояний, которые могут быть преобразованы в эквивалентные распознаватели с магазинной памятью и одним состоянием без увеличения мощности множества магазинных символов. Приводятся их формальные описания и на их основе – правила выполнения преобразования. Представлен пример преобразования распознавателя с конечным числом состояний в распознаватель с одним состоянием. Приводятся протоколы работы распознавателей при обработке входной цепочки, подтверждающие правильность выполненных преобразований.

Распознаватель с одним состоянием в процессе распознавания анализирует только входную цепочку и содержимое магазина. Это позволяет сократить количество параметров, определяющих поведение распознавателя с магазинной памятью. Распознаватель с одним состоянием имеет более компактное представление, чем распознаватель с конечным множеством состояний.

Ключевые слова: *контекстно-свободный язык, распознаватель с магазинной памятью, состояние, эквивалентные преобразования.*

Одной из важных задач обработки формальных языков является задача распознавания, которая заключается в определении принадлежности заданной цепочки заданному языку. Для решения задачи распознавания контекстно-свободных языков используются распознаватели с магазинной памятью (МП–распознаватели) [1–7]. МП-распознаватель можно представить устройством, изображенным на рис. 1.

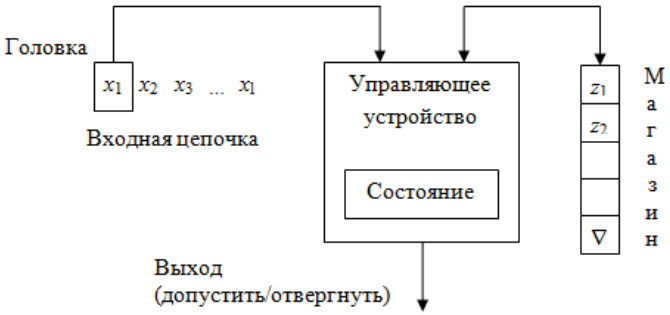

Рис. 1. МП-распознаватель

В работе [8] представлен алгоритм синтеза МП–распознавателей с конечным числом состояний, которые формально можно представить следующим образом:

$$МП^n = (Q^n, \Sigma^n, I^n, I^n, S^n, P^n, E^n, \delta^n, \lambda^n, q_0^n, \boldsymbol{q}^n, \gamma_0^n),$$

где Q^n – конечное множество состояний, $Q^n = \{q_0^n, q_1^n, \ldots, q_m^n\}$; Σ^n – конечное множество входных символов, включающее концевой маркер ⊣, которым заканчивается входная цепочка; $I^n = Q^n \cup \{\nabla\}$ – конечное множество магазинных символов (равно множеству состояний, дополненному маркером дна магазина ∇); I^n – конечное множество операций над головкой, I^n = (*сдвиг*, *держать*). Операция *сдвиг* перемещает головку на одну позицию вправо, а *держать* – не изменяет положения головки; S^n – конечное множество операций над состоянием, $S^n = \{сост(q_0^n), сост(q_1^n), \ldots, сост(q_m^n)\}$. Операция $сост(q_i^n)$ обозначает переход в состояние q_i^n; P^n – множество операций над магазином, $P^n = \{зам(\gamma_1), зам(\gamma_2), \ldots зам(\gamma_i), \ldots\}$. Операция $зам(\gamma_i)$ заключается в выталкивании верхнего символа из магазина и последовательном вталкивании симво-

лов цепочки γ_i; E^n – конечное множество значений выхода, E^n = {*допустить*, *отвергнуть*}; q_0^n – начальное состояние; $\boldsymbol{q}^n$ – допускающее состояние; γ_0^n – начальное содержимое магазина, $\gamma_0^n = \nabla$ (магазин пуст);

$\delta^n : Q^n \times \Sigma^n \times \Gamma^n \to I^n \times S^n \times P^n$ – частичная функция переходов, которая состоянию, символу входной цепочки (находящемуся под головкой) и верхнему символу магазина ставит в соответствие операцию над головкой, состоянием и магазином, причем множество видов значений на тройке (q, a, x) равно {(*сдвиг*, *сост*(p), *зам*(x)), (*держать*, *сост*(p), *зам*(xr)), (*держать*, *сост*(x), *зам*(ε))}, где ε – пустая цепочка. Заметим, что на тройке (q, a, x) операция *зам*(x) не изменяет содержимого магазина, *зам*(xr) – добавляет один символ в магазин, а *зам*(ε) – выталкивает верхний символ из магазина.

$\lambda^n : Q^n \times \Sigma^n \times \Gamma^n \to E^n$ – частичная функция выходов, которая состоянию, символу входной цепочки (находящемуся под головкой) и верхнему символу магазина ставит в соответствие значение выхода – *допустить* или *отвергнуть*. Значение функции на тройке ($\boldsymbol{q}$, ┤, ∇) равно *допустить*, а на всех остальных, на которых функция определена — *отвергнуть*.

Области определения функций δ^n и λ^n не пересекаются, а их объединение равно области отправления.

Тройка (q, α, γ), где q – состояние, α – часть входной цепочки, начиная с символа под головкой и заканчивая концевым маркером, γ – содержимое магазина, называется конфигурацией распознавателя $МП^n$. Исходной конфигурацией является (q_0^n, α_0, ∇), где α_0 – вся входная цепочка (головка находится над первым символом).

Пусть конфигурацией $МП^n$ является тройка (q, $a\alpha$, $x\gamma$), где a – символ под головкой, x – верхний символ магазина. Если на тройке (q, a, x) определена функция переходов δ^n, то ее значение определяет операции над головкой, состоянием и магазином. При выполнении этих операций конфигурация изменяется. Если на тройке (q, a, x) определена функция выходов λ^n, то процесс распознавания заканчивается с результатом, равным значению функции λ^n. Такую конфигурацию назовем заключительной. Итак, работа $МП^n$ заключается в изменении конфигураций. Последней является заключительная конфигурация, в которой определяется результат распознавания.

Покажем, что распознаватель $МП^n$ можно преобразовать в распознаватель $МП^1$ с одним состоянием, который распознает тот же язык, что и $МП^n$. Формально $МП^1$ определим следующим образом:

$$МП^1 = (\Sigma^1, \Gamma^1, I^1, P^1, E^1, \delta^1, \lambda^1, \gamma_0^1),$$

где $\Sigma^1 = \Sigma^n$, $\Gamma^1 = \Gamma^n$, $I^1 = I^n$, $P^1 = P^n$, $E^1 = E^n$.

В $МП^1$ только одно состояние, поэтому операция над состоянием не имеет смысла, функции переходов δ^1 и выходов λ^1 определяются как

$\delta^1 : \Sigma^1 \times \Gamma^1 \to I^1 \times P^1$ и $\lambda^1 : \Sigma^1 \times \Gamma^1 \to E^1$, а конфигурацией является двойка (α, γ).

Роль состояния в $МП^1$ будет играть верхний символ магазина, поэтому конфигурации (q, α, $x\gamma$) в $МП^n$ будет соответствовать конфигурация (α, $qx\gamma$) в $МП^1$. Исходной конфигурации (q_0^n, α_0, ∇) распознавателя $МП^n$ соответствует конфигурация (α_0, $q_0^n\nabla$) в $МП^1$, поэтому начальным содержимым магазина в $МП^1$ будет $q_0^n\nabla$. Определим функцию переходов δ^1 так, что если на i-ом шаге обработки входной цепочки $МП^n$ находится в конфигурации (q, $a\alpha$, $x\gamma$), то $МП^1$ на этом же шаге находится в конфигурации ($a\alpha$, $qx\gamma$).

Пусть в конфигурации (q, $a\alpha$, $x\gamma$) определена функция переходов δ^n, тогда в конфигурации ($a\alpha$, $qx\gamma$) должна быть определена функция переходов δ^1.

Если δ^n (q, a, x) = (*сдвиг*, *сост*(p), *зам*(x)), то на i+1-ом шаге конфигурацией $МП^n$ будет (p, α, $x\gamma$), а ей в $МП^1$ соответствует конфигурация (α, $px\gamma$). Распознаватель $МП^1$ сменит конфигурацию ($a\alpha$, $qx\gamma$) на (α, $px\gamma$), если δ^1 (a, q) = (*сдвиг*, *зам*(p)).

Если δ^n (q, a, x) = (*держать*, *сост*(p), *зам*(xr)), то на i+1-ом шаге конфигурацией $МП^n$ будет (p, $a\alpha$, $rx\gamma$), а ей в $МП^1$ соответствует конфигурация ($a\alpha$, $prx\gamma$). Распознаватель $МП^1$ сменит конфигурацию ($a\alpha$, $qx\gamma$) на ($a\alpha$, $px\gamma$), если δ^1 (a, q) = (*держать*, *зам*(rp)).

Если δ^n (q, a, x) = (*держать*, *сост*(x), *зам*(ε)), то на i+1-ом шаге конфигурацией $МП^n$ будет (x, $a\alpha$, γ), а ей в $МП^1$ соответствует конфигурация ($a\alpha$, $x\gamma$). Распознаватель $МП^1$ сменит конфигурацию ($a\alpha$, $qx\gamma$) на ($a\alpha$, $x\gamma$), если δ^1 (a, q) = (*держать*, *зам*(ε)).

Если в конфигурации (q, $a\alpha$, $x\gamma$) распознавателя $МП^n$ определена функция выходов λ^n и λ^n (q, a, x) = *отвергнуть*, тогда в конфигурации ($a\alpha$, $qx\gamma$) распознавателя $МП^1$ должна быть определена функция выходов λ^1 и λ^1 (a, x) = *отвергнуть*.

Рассмотрим конфигурацию ($\boldsymbol{q}$, ┤, ∇) распознавателя $МП^n$, на которой определена функция выходов λ^n и λ^n (q, a, x) = *допустить*. Этой конфигурации в $МП^1$ соответствует конфигурация (┤, $\boldsymbol{q}\nabla$) в $МП^1$, т. е. входная цепочка закончилась и в магазине только допускающее состояние. Для того, чтобы убедиться в том, что в магазине действительно только допускающее состояние, вытолкнем его из магазина (δ^1 ((┤, $\boldsymbol{q}$) = (*держать*, *зам*(ε))) и получим конфигурацию (┤, ∇), в которой функция выходов λ^1 равна *допустить* (λ^1 (┤, ∇) = *допустить*).

Таким образом, описаны правила преобразования МП-распознавателя с конечным множеством состояний в эквивалентный ему МП-распознаватель с одним состоянием.

Рассмотрим пример выполнения преобразования. МП-распознаватель с конечным множеством состояний можно задать таблицей (табл. 1), состоящей из четырех столбцов. В первом столбце указывается состояние, во втором – множество входных символов, в третьем – магазинный символ или пусто. Если МП-распознаватель находится в конфигурации (q, $a\alpha$, $x\gamma$) и в таблице есть строка, в которой в первом элементе (столбце) записано состояние q, во втором – множество, содержащее символ a, в третьем – символ x или пусто, то в четвертом столбце записаны действия, которые должен выполнить распознаватель. Для сокращения таблицы в четвертом столбце не указывается операция над головкой *держать*, которая не изменяет положения головки, не указывается операция *зам*(x), которая не изменяет содержимого магазина, операция *зам*(ε) записывается как *вытолкнуть*, а операция *зам*(xr) – как *втолкнуть*(r). Если же МП-распознаватель находится в конфигурации (q, $a\alpha$, $x\gamma$) и в таблице нет строки, в которой в первом элементе (столбце) записано состояние q, во втором – множество, содержащее символ a, в третьем – символ x или пусто, то цепочка отвергается.

В МП-распознавателе, представленном в табл. 1, состояние 1 – начальное, состояние 4 – допускающее, начальное содержимое магазина – магазин пуст.

В табл. 2 представлен протокол работы МП-распознавателя (табл. 1) при обработке цепочки *adedc*┤.

Таблица 1

МП-распознаватель с конечным множеством состояний

Текущее состояние	Входные символы	Верх магазина	Действия
1	*a*		*сдвиг, сост*(3)
1	*b, c, d, e*		*сост*(5), *втолкнуть*(2)
2	*c*		*сдвиг, сост*(4)
3	*d, e*		*сост*(9), *втолкнуть*(4)
4	*d, e*		*сост*(9), *втолкнуть*(2)
4	┤	∇	*допустить*
5	*b*		*сдвиг, сост*(6)
5	*c*	2	*сост*(2), *вытолкнуть*
5	*d, e*		*сост*(9), *втолкнуть*(7)
6	*d, e*		*сост*(9), *втолкнуть*(8)
7	*d*		*сдвиг, сост*(8)
8	*c*	2	*сост*(2), *вытолкнуть*
8	*a*		*сдвиг, сост*(5)
9	*e*		*сдвиг, сост*(10)
9	*d*		*сдвиг, сост*(11)
10	*d, e*		*сост*(9), *втолкнуть*(11)
11	*a, c, d, e,* ┤	2	*сост*(2), *вытолкнуть*
11	*a, c, d, e,* ┤	4	*сост*(4), *вытолкнуть*
11	*a, c, d, e,* ┤	7	*сост*(7), *вытолкнуть*
11	*a, c, d, e,* ┤	8	*сост*(8), *вытолкнуть*
11	*a, c, d, e,* ┤	11	*сост*(11), *вытолкнуть*

Таблица 2

Протокол работы МП-распознавателя

Шаг	Состояние	Символ	Магазин	Действие
1	1	*a*	∇	*сдвиг, сост*(3)
2	3	*d*	∇	*сост*(9), *втолкнуть*(4)
3	9	*d*	4 ∇	*сдвиг, сост*(11)
4	11	*e*	4 ∇	*сост*(4), *вытолкнуть*
5	4	*e*	∇	*сост*(9), *втолкнуть*(2)
6	9	*e*	2 ∇	*сдвиг, сост*(10)
7	10	*d*	2 ∇	*сост*(9), *втолкнуть*(11)
8	9	*d*	11 2∇	*сдвиг, сост*(11)
9	11	*c*	11 2 ∇	*сост*(11), *вытолкнуть*
10	11	*c*	2 ∇	*сост*(2), *вытолкнуть*
11	2	*c*	∇	*сдвиг, сост*(4)
12	4	┤	∇	*допустить*

В результате выполнения преобразования получим МП-распознаватель с одним состоянием, который можно задать таблицей (табл. 3), строки которой соответствуют магазинным символам и маркеру дна, а столбцы – входным символам и концевому маркеру. В клетке таблицы, находящейся в строке *x* и столбце *a*, записывается значение функции перехода или выхода на паре (*a*, *x*). Для того, чтобы не загромождать таблицу, операции *держать* и *отвергнуть* записывать не будем, а операцию *зам*(ε) будем записывать как *вытолкнуть*. Начальным содержимым магазина МП-распознавателя (табл. 3) будет 1 ∇.

Таблица 3

МП-распознаватель с одним состоянием

	a	*b*	*c*	*d*	*e*	┤
1	*зам* (3) *сдвиг*	*зам* (2 5)	*зам* (2 5)	*зам* (2 5)	*зам* (2 5)	
2			*зам* (4) *сдвиг*			
3				*зам* (4 9)	*зам* (4 9)	
4				*зам* (2 9)	*зам* (2 9)	*вытолкнуть*
5		*зам* (6) *сдвиг*	*вытолкнуть*	*зам* (7 9)	*зам* (7 9)	
6				*зам* (8 9	*зам* (8 9)	
7				*зам* (8)*сдвиг*		
8	*зам* (5)*сдвиг*		*вытолкнуть*			
9				*зам* (11)*сдвиг*	*зам* (10)*сдвиг*	
10				*зам* (11 9)	*зам* (11 9)	
11	*вытолкнуть*		*вытолкнуть*	*вытолкнуть*	*вытолкнуть*	*вытолкнуть*
∇						*допустить*

В табл. 4 представлен протокол работы МП-распознавателя (табл. 3) при обработке цепочки *adedc*┤. Сравнивая протоколы работы распознавателей можно сделать вывод о том, что на каждом шаге работы содержимое магазина МП-распознавателя с одним состоянием отличается от содержимого магазина МП-распознавателя с множеством состояний на соответствующем шаге только наличием в вершине магазина текущего состояния МП-распознавателя с множеством состояний.

Таким образом, в статье описаны правила преобразования МП-распознавателя с конечным множеством состояний в эквивалентный ему МП-распознаватель с одним состоянием, который в процессе распознавания анализирует только входную цепочку и содержимое магазина. Это позволяет сократить количество параметров, определяющих поведение распознавателя с магазинной памятью. Распознаватель с одним состоянием имеет более компактное представление, чем распознаватель с конечным множеством состояний. При этом, следует отметить, устранение множества состояний не приводит к расширению множества магазинных символов.

Таблица 4

Протокол работы МП-распознавателя с одним состоянием

Шаг	Символ	Магазин	Действие
1	*a*	1 ∇	ЗАМ(3) СДВИГ
2	*d*	3 ∇	ЗАМ(4 9) ДЕРЖАТЬ
3	*d*	9 4 ∇	ЗАМ(11) СДВИГ
4	*e*	11 4 ∇	ВЫТОЛК ДЕРЖАТЬ
5	*e*	4 ∇	ЗАМ(2 9) ДЕРЖАТЬ
6	*e*	9 2 ∇	ЗАМ(10) СДВИГ
7	*d*	10 2 ∇	ЗАМ(11 9) ДЕРЖАТЬ
8	*d*	9 11 2 ∇	ЗАМ(11) СДВИГ
9	*c*	11 11 2∇	ВЫТОЛК ДЕРЖАТЬ
10	*c*	11 2 ∇	ВЫТОЛК ДЕРЖАТЬ
11	*c*	2 ∇	ЗАМ(4) СДВИГ
12	┤	4 ∇	ВЫТОЛК ДЕРЖАТЬ
13	┤	∇	ДОПУСТИТЬ

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Schutzenberger M.P. «On context-free languages and pushdown automata», Information and Control 6:3 (1963). pp. 246. 264.

2. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. М.: Мир. 1978. Т. 1. 612 с.

3. Льюис Ф., Розенкранц Д., Стирнз Р. Теоретические основы проектирования компиляторов. М. : Мир, 1979. 656 с.

4. Опалева Э.А., Самойленко В.П. Языки программирования и методы трансляции. СПб.: «БХВ-Петербург», 2005. 471 с.

5. Ахо А., Лам М., Сети Р., Ульман Дж. Компиляторы. Принципы, технологии и инструментарий. М: Издательский дом «Вильямс», 2008. 1185 с.

6. Серебряков, В.А. Теория и реализация языков программирования. М.: Физматлит, 2012. 233 с.

7. Поляков В.М., Рязанов Ю. Д. Алгоритм построения нерекурсивных программ-распознавателей линейной сложности по детерминированным синтаксическим диаграммам // Вестник БГТУ им. В.Г. Шухова. № 6. 2013. С. 194–199.

8. Рязанов Ю. Д. Синтез распознавателей с магазинной памятью по детерминированным синтаксическим диаграммам // Вестник ВГУ. Системный анализ и информационные технологии. 2014. №1. С. 138–145.

Ryazanov Yu.D.

TRANSFORMING OF THE PUSHDOWN RECOGNIZER WITH FINITE SET OF STATES INTO RECOGNIZER WITH ONE STATE

In this article the recognition of the problem of context-free languages is considered. Pushdown recognizers are used for its decision, which can have a finite set of states. In this work the class of pushdown recognizers and a finite set of states which can be transformed to equivalent recognizers with pushdown memory and one state without increase in power of a set of pushdown symbols. Their formal descriptions are given and on their basis rules of performance of transformation are provided. The example of transformation of the recognizer with final number of states to the recognizer with one state is presented. The records recognizers work at processing an input string, are given validating the executed transformations.

The recognizer with one state in the course of recognition analyzes only the input string and the contents of the pushdown memory. It allows to reduce the number of parameters defining behavior of the recognizer with pushdown memory. The recognizer with one state has more compact idea, than the recognizer with a final set of states.

***Key words**: context-free language, pushdown recognizer, state, equivalent transforming.*

Рязанов Юрий Дмитриевич, магистрант кафедры программного обеспечения вычислительной техники и автоматизированных систем.

Белгородский государственный технологический университет им. В.Г. Шухова.

Адрес: Россия, 308012, Белгород, ул. Костюкова, д. 46.

E-mail: Ryazanov.iurij@yandex.ru