Конечные автоматы

Действовать без правил – самое трудное и самое утомительное занятие на этом свете.

А. Мандзони

Содержание

- Понятие конечного автомата
- Приведение конечного автомата
- Недетерминированный конечный автомат
- Реализация конечного автомата

Понятие конечного автомата

- *Конечный автомат* – модель вычислительного устройства с фиксированным и конечным объемом памяти, которое читает и обрабатывает цепочку входных символов, принадлежащих некоторому конечному множеству.
- Конечные автоматы различают в зависимости от того, какой результат они дают на выходе.

Конечный распознаватель

- *Конечный распознаватель* – модель устройства с конечным числом состояний, которое отличает *допустимые* (правильно образованные) цепочки от *недопустимых*.
- *Регулярное множество* конечного распознавателя – множество его допустимых цепочек.
- Задание конечного распознавателя:
  - *Входной алфавит*: A=$\{a_1, \ldots, a_M\}$
  - *Множество состояний*: S=$\{S_1, \ldots, S_N\}$
  - *Функция перехода*: $\delta(S_i,a_j)=S_k \ \forall \ i,k \in 1..N, j \in 1..M$
  - *Начальное состояние*
  - Подмножество *допускающих* и *заключительных* состояний; подмножество *отвергающих* состояний.

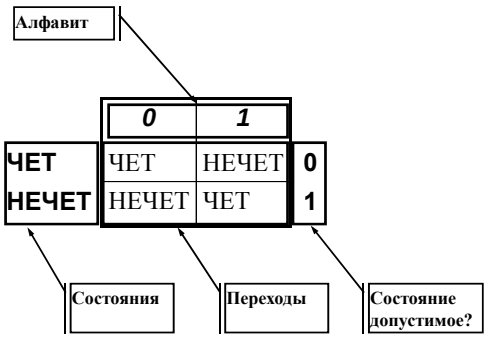
Пример конечного распознавателя

- «Контролер нечетности единиц» с регулярным множеством всех цепочек, состоящих из 0 и 1 и имеющих нечетное число единиц.
- Задание распознающего автомата:
  - Алфавит: {0, 1}
  - Состояния: {ЧЕТ, НЕЧЕТ}
  - Начальное состояние: ЧЕТ
  - Функция перехода:
    δ(ЧЕТ,0)=ЧЕТ
    δ(ЧЕТ,1)=НЕЧЕТ
    δ(НЕЧЕТ,0)=НЕЧЕТ
    δ(НЕЧЕТ,1)=ЧЕТ
  - Допускающие состояния: {НЕЧЕТ}

Пример конечного распознавателя

Цепочка	Переходы	Результат
1101	ЧЕТ$^{(1)}\rightarrow$НЕЧЕТ$^{(1)}\rightarrow$ЧЕТ$^{(0)}\rightarrow$ЧЕТ$^{(1)}\rightarrow$ НЕЧЕТ	Допустить
101	ЧЕТ$^{(1)}\rightarrow$НЕЧЕТ$^{(0)}\rightarrow$НЕЧЕТ$^{(1)}\rightarrow$ЧЕТ	Отвергнуть
11111	ЧЕТ$^{(1)}\rightarrow$НЕЧЕТ$^{(1)}\rightarrow$ЧЕТ$^{(1)}\rightarrow$НЕЧЕТ$^{(1)}$ $\rightarrow$ЧЕТ$^{(1)}\rightarrow$НЕЧЕТ	Допустить
1	ЧЕТ$^{(1)}\rightarrow$НЕЧЕТ	Допустить
0	ЧЕТ$^{(0)}\rightarrow$ЧЕТ	Отвергнуть

Таблица переходов



Пример таблицы переходов

	a	*b*	*c*	
1	1	3	4	**1**
2	2	1	3	**0**
3	2	4	4	**1**
4	3	3	3	**0**

- Алфавит:
  {a, b, c}
- Состояния:
  {1, 2, 3, 4}
- Начальное состояние:
  1
- Допускающие состояния:
  {1, 3}
- Примеры цепочек:
  abcc – допустимая
  cba – недопустимая

Обрабатывающий автомат

- *Обрабатывающий автомат* – конечный распознаватель, который прекращает свою работу в одном из двух случаев:
  1) обнаружение ошибки во входной цепочке
  2) исчерпание входной цепочки.
- Обрабатывающий автомат можно построить из соответствующего распознающего автомата, если:
  1) дополнить множество состояний автомата *состоянием ошибки* **E**.
  2) дополнить алфавит автомата *концевым маркером* **–|**.

Пример обрабатывающего автомата

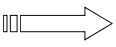
- «Контролер парности единиц» с регулярным множеством всех цепочек, состоящих из 0 и 1, в которых единицы либо отсутствуют, либо встречаются парами.
- *Распознающий* автомат
  (после перехода в состояние E может выполнять «лишнюю» работу):

	0	***1***	
1	1	2	**1**
2	E	1	**0**
E	E	E	**0**

Пример обрабатывающего автомата

- *Обрабатывающий* автомат
  (добавлен концевой маркер и вызовы подпрограмм обработки состояний «ДА» и «НЕТ»):

	0	***1***	⊣
1	1	2	ДА
2	E	1	НЕТ
E	E	E	НЕТ



	0	***1***	⊣
1	1	2	ДА
2	НЕТ	1	НЕТ

Метод разметки символов

- Строим конечный распознаватель выражения:

<выражение>::=<операнд><знак><операнд>
<операнд>::=<идентификатор> | <элемент массива>
<элемент массива>::=<идентификатор>**[**<индекс>**]**
<индекс>::=<целое без знака> | <идентификатор>
{, <индекс>}
<знак>::=**+** | **-** | ***** | **/**

Допустимые цепочки:
A[1]+B,
A[1, j, 3]*B

Метод разметки символов

- Предполагаем, что некоторый другой конечный автомат предварительно распознает цепочку и выдает следующие лексемы:

  ID идентификатор
  INT целое без знака
  S знак операции
  [левая скобка
  ] правая скобка
  , запятая

  Будем считать их алфавитом нашего автомата.

Метод разметки символов

- Возьмем типичное выражение и разметим его (назначим символам, имеющим одинаковую семантику, одинаковые метки):

	A	[	1	,	j	,	3	]	*	B
НАЧ	ИД	ЛСК	ЦЕЛ	ЗПТ	ИДКС	ЗПТ	ЦЕЛ	ПСК	ЗНАК	ИД

Метод разметки символов

- Построим конечный распознаватель:

	ID	*INT*	*s*	*[*	*]*	*,*	
НАЧ	ИД	Е	Е	Е	Е	Е	**0**
ИД	Е	Е	ЗНАК	ЛСК	Е	Е	**1**
ЛСК	ИДКС	ЦЕЛ	Е	Е	Е	Е	**0**
ЦЕЛ	Е	Е	Е	Е	ПСК	ЗПТ	**0**
ЗПТ	ИДКС	ЦЕЛ	Е	Е	Е	Е	**0**
ИДКС	Е	Е	Е	Е	ПСК	ЗПТ	**0**
ПСК	Е	Е	ЗНАК	Е	Е	Е	**1**
ЗНАК	ИД	Е	Е	Е	Е	Е	**0**
Е	Е	Е	Е	Е	Е	Е	**0**

Метод разметки символов

- Построенный распознаватель *не оптимален*:

	ID	*INT*	*s*	*[*	*]*	*,*	
НАЧ	ИД	Е	Е	Е	Е	Е	**0**
ИД	Е	Е	ЗНАК	ЛСК	Е	Е	**1**
ЛСК	ИДКС	ЦЕЛ	Е	Е	Е	Е	**0**
ЦЕЛ	Е	Е	Е	Е	ПСК	ЗПТ	**0**
ЗПТ	ИДКС	ЦЕЛ	Е	Е	Е	Е	**0**
ИДКС	Е	Е	Е	Е	ПСК	ЗПТ	**0**
ПСК	Е	Е	ЗНАК	Е	Е	Е	**1**
ЗНАК	ИД	Е	Е	Е	Е	Е	**0**
Е	Е	Е	Е	Е	Е	Е	**0**

Замечание о пустой цепочке

- *Пустая цепочка* ε – цепочка, не содержащая ни одного символа.
- ε допускается автоматом $\Leftrightarrow$ начальное состояние автомата допустимо.
- ε не является входным символом автомата.
- ε не совпадает с пустым множеством { } и пробелом « ».

Приведение конечного автомата

- Утверждение 1:
  Для любого конечного автомата имеется бесконечное множество конечных автоматов с *совпадающим регулярным множеством цепочек и отличным* от исходного *числом состояний*.
- Утверждение 2:
  Для любого конечного распознавателя имеется **единственный** конечный распознаватель с *совпадающим регулярным множеством цепочек* и *не превышающим исходное* числом состояний, называемый *минимальным*.

Редукция конечного автомата

- *Редукция конечного автомата* – процесс приведения данного автомата к минимальному.
- Этапы редукции:
  1. Замена нескольких эквивалентных состояний на одно.
  2. Отбрасывание недостижимых состояний.

Эквивалентные состояния

- Определение 1:
  Состояние ***s*** конечного распознавателя ***M*** эквивалентно состоянию ***t*** конечного распознавателя ***N***, если ***M***, начав работу в состоянии ***s***, будет допускать в точности те же цепочки, что и ***N***, начавший работу в состоянии ***t***.
- Пример:

	a	*b*	
1	1	4	**0**
2	3	5	**1**
3	5	0	**0**
4	2	3	**1**
5	2	3	**1**

	a	*b*	
1	1	**X**	**0**
2	3	**X**	**1**
3	**X**	1	**0**
X	2	3	**1**
X	2	3	**1**

	a	*b*	
1	1	X	**0**
2	3	X	**1**
3	X	1	**0**
X	2	3	**1**

Эквивалентные состояния

- Определение 2:
  Состояние ***s*** конечного распознавателя ***M*** эквивалентно состоянию ***t*** конечного распознавателя ***N***, если для ***s*** и ***t*** не существует *различающей* цепочки – под действием которой состояние ***s*** переходит в допускающее состояние, а состояние ***t*** – в отвергающее состояние (или наоборот).
- Пример:

M

	0	*1*	
A	A	C	**0**
B	B	C	**0**
C	B	A	**1**

N

	0	*1*	
X	X	Y	**0**
Y	Z	X	**1**
Z	X	Z	**0**

Состояния **A** и **X** не эквивалентны, так как
101 – различающая их цепочка

Эквивалентные автоматы

- Два конечных автомата эквивалентны, если эквивалентны их начальные состояния.
- Замечание. Понятие эквивалентности автоматов является отношением эквивалентности (выполняются свойства рефлексивности, симметричности и транзитивности). *Докажите!*
- Пример:

M

	0	*1*	
A	A	C	**0**
B	B	C	**0**
C	B	A	**1**

N

	0	*1*	
X	X	Y	**0**
Y	Z	X	**1**
Z	X	Z	**0**

Автоматы **M** и **N** не эквивалентны, так как не эквивалентны их начальные состояния **A** и **X**.

Признак эквивалентности состояний

- Два состояния эквивалентны ⇔ выполняются
  1. *условие подобия*:
     состояния оба допускающие или оба отвергающие
  2. *условие преемственности*:
     преемники данных состояний эквивалентны (для любых входных символов данные состояния переходят в эквивалентные состояния).
- Замечание. Данные условия выполняются ⇔ состояния не имеют различающей цепочки. *Докажите!*

Пример поиска эквивалентных состояний

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

1. Проверим состояния 0 и 7:
   - условие подобия – выполняется;
   - условие преемственности – не выполняется:

	a	*b*	
0,7	0,6	**0,1**	3,3	**0,0**

Таблица эквивалентности

Нарушено условие подобия для состояний-преемников

Состояния 0 и 7 не эквивалентны, ***a*** – различающая их цепочка.

Пример поиска эквивалентных состояний

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

2. Проверим состояния 0 и 1:
   - условие подобия – выполняется;
   - условие преемственности –

	a		*b*	
0,1	0,2	**0,0**	3,5	**0,0**

⇒

	a		*b*	
0,1	0,2	**0,0**	3,5	**0,0**
0,2	0,2	**0,0**	3,7	**0,0**
3,5	6		7,5	**0,0**
3,7	6		3,7	**0,0**
7,5	6		3,5	**0,0**

⇒ Состояния 0 и 1 эквивалентны, так как не существует различающая их цепочка.

⇒ Состояния 0 и 2, 3 и 5, 3 и 7 эквивалентны.

Пример поиска эквивалентных состояний

3. Состояния 0 и 1, 0 и 2, 3 и 5, 3 и 7 эквивалентны
   ⇒ **X**≅0≅1≅2,
   Y≅3≅5≅7.

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

⇒

	a	*b*	
X	**X**	**Y**	**0**
Y	6	**Y**	**0**
4	**X**	6	**1**
6	6	**Y**	**1**

Недостижимые состояния

- *Недостижимое состояние* – состояние, которое не достигается из начального ни для какой входной цепочки.
  Строки недостижимых состояний таблицы переходов можно удалить, получая эквивалентный конечный автомат.
- Алгоритм поиска недостижимых состояний:
  1. Найти список всех *достижимых* состояний:
     1) Занести в список начальное состояние
     2) Для каждого состояния из списка занести в него все его состояния-преемники, которые пока отсутствуют в списке.
  2. Найти разность всего множества состояний и списка достижимых состояний.

Редукция автомата методом разбиения

- Метод таблиц эквивалентности не эффективен, так как обрабатывает одновременно только два состояния.
- *Метод разбиения* заключается в последовательном разбиении множества состояний на такие непересекающиеся подмножества (*блоки*), что неэквивалентные состояния будут попадать в *разные* подмножества.

Пример редукции автомата методом разбиения

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

1. Разобьем состояния на два блока – «отвергающие» и «допускающие»: $P_0=(\{1,2,3,4\}, \{5,6,7\})$
2. Разобьем блок {1,2,3,4} из P_0 относительно символа ***a***: $\{1,2\}^{(a)}\rightarrow\{6,7\}\subset\{5,6,7\}$, $\{3,4\}^{(a)}\rightarrow\{1,4\}\subset\{1,2,3,4\}$. Состояния-преемники {1,2} и {3,4} не эквивалентны по входу ***a***. $P_0\rightarrow P_1=(\{1,2\}, \{3,4\}, \{5,6,7\})$.

Пример редукции автомата методом разбиения

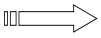
	a	*b*	
1	6	3	0
2	7	3	0
3	1	5	0
4	4	6	0
5	7	3	1
6	4	1	1
7	4	2	1

3. Разобьем блок {3,4} из P_1 относительно символа ***a***: $3^{(a)}\rightarrow 1\subset\{1,2\}$, $4^{(a)}\rightarrow 4\subset\{3,4\}$. Состояния-преемники 3 и 4 не эквивалентны по входу ***a***. $P_1\rightarrow P_2=(\{1,2\}, \{3\}, \{4\}, \{5,6,7\})$.
4. Разобьем блок {5,6,7} из P_2 относительно символа ***b***: $5^{(b)}\rightarrow 3\subset\{3\}$, $\{6,7\}^{(b)}\rightarrow 1\subset\{1,2\}$. Состояния-преемники {5} и {6,7} не эквивалентны по входу ***a***. $P_2\rightarrow P_3=(\{1,2\}, \{3\}, \{4\}, \{5\}, \{6,7\})$.

Пример редукции автомата методом разбиения

5. **P_3 =({1,2}, {3}, {4}, {5}, {6,7})**
   не допускает дальнейшего разбиения:
   $\{1,2\}^{(a)}\rightarrow\{6,7\}$, $\{1,2\}^{(b)}\rightarrow\{3\}$,
   $\{6,7\}^{(a)}\rightarrow\{4\}$, $\{6,7\}^{(b)}\rightarrow\{1,2\}$.
   Состояния внутри каждого блока из P_3 эквивалентны.

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



	a	*b*	
X	Y	3	0
3	X	5	0
4	4	Y	0
5	Y	3	1
Y	4	X	1

Недетерминированный конечный автомат

- *Недетерминированный конечный автомат* – конечный автомат, у которого значением функции перехода является *множество* состояний.
- Задание недетерминированного конечного распознавателя:
  - *Входной алфавит*: A=$\{a_1, \ldots, a_M\}$
  - *Множество состояний*: S=$\{S_1, \ldots, S_N\}$
  - *Функция перехода*: $\delta(S_i,a_j) = \{Sk_1, \ldots, Sk_N\}\ \forall\ i,k\in 1..N, j\in 1..M$
  - *Подмножество начальных состояний*
  - Подмножество *допускающих* состояний.
- Недетерминированный конечный распознаватель допускает входную цепочку, если она позволяет связать *одно* из его начальных состояний с *одним* из допускающих состояний.

Пример недетерминированного автомата

	0	*1*	
→**A**	A,B	C	**0**
→**B**	B	C	**1**
C		A,C	**1**

- Алфавит: {0,1}
- Состояния: {A,B,C}
- Начальные состояния: {A,B}
- Переходы:
  δ(A,0)={A,B}, δ(A,1)={C},
  δ(B,0)={B}, δ(B,1)={C},
  δ(C,0)={}, δ(C,1)={A,C}.
  Переход в пустое множество состояний означает, что дальнейшие переходы невозможны и входная цепочка отвергается.
- Допускающие состояния: {B,C}
- Допустимые цепочки: 11, 011, 000
  Недопустимые цепочки: 10, 010

Пример недетерминированного распознавателя

- Семантика: распознавание цепочек ЗАДАЧА, ЗАЧЁТ
- Алфавит: {А, Д, Ё, З, Т, Ч}
- Состояния:
  нач – начальное
  $З_1$ – З в Задача
  $З_2$ – З в Зачёт
  $А_1$ – первая А в Задача
  $А_2$ – вторая А в Задача
  $А_3$ – третья А в Задача
  $А_4$ – А в Зачёт
  Д – Д в Задача
  $Ч_1$ – Ч в Задача
  $Ч_2$ – Ч в Зачёт
  Ё – Е в Зачёт
  Т – Т в Зачёт
- Начальные состояния: {нач}

Пример недетерминированного распознавателя

	А	*Д*	*Ё*	*З*	*Т*	*Ч*	
→нач				$З_1, З_2$			0
$З_1$	$А_1, А_4$						0
$З_2$	$А_1, А_4$						0
$А_1$		Д					0
$А_2$						$Ч_1$	0
$А_3$							1
$А_4$						$Ч_2$	0
Д	А2						0
$Ч_1$	А3						0
$Ч_2$			Ё				0
Ё					Т		0
Т							1

Эквивалентность недетерминированных и детерминированных распознавателей

- Для любого
  недетерминированного конечного распознавателя
  существует
  эквивалентный ему
  детерминированный конечный распознаватель.

Алгоритм построения эквивалентного детерминированного автомата

N

	0	*1*	
→**A**	A,B	C	**0**
→**B**	B	C	**1**
C		A,C	**1**

⇩

D

	0	*1*
{A, B}		

1. Пометить 1-ю строку таблицы переходов автомата D множеством начальных состояний автомата N.

Алгоритм построения эквивалентного детерминированного автомата

N

	0	*1*	
→**A**	A,B	C	**0**
→**B**	B	C	**1**
C		A,C	**1**

⇩

D

	0	*1*
{A, B}	{A,B}	{C}

2. По множеству состояний *S*, которое помечает строку таблицы переходов автомата D, вычислить те состояния автомата N, которые могут быть достигнуты из *S* с помощью каждого символа алфавита, и поместить их в соответствующие ячейки таблицы переходов автомата N.

Алгоритм построения эквивалентного детерминированного автомата

N

	0	*1*	
→**A**	A,B	C	**0**
→**B**	B	C	**1**
C		A,C	**1**

⇩

D

	0	*1*
{A, B}	{A,B}	{C}
{C}		

3. Для каждого множества состояний, полученного на шаге 2., проверить, есть ли в таблице переходов автомата D строка, помеченная этим множеством.
   Если есть, то добавить в таблицу переходов автомата D новую строку и пометить ее этим множеством.

Алгоритм построения эквивалентного детерминированного автомата

4. Для всех строк в таблице переходов автомата D, для которых еще не вычислены переходы, применить шаг 2.

D

	0	*1*
{A, B}	{A,B}	{C}
{C}	{ }	{A,C}

⇒

D

	0	*1*
{A, B}	{A,B}	{C}
{C}	{}	{A,C}
{ }		
{A,C}		

Алгоритм построения эквивалентного детерминированного автомата

4. Для всех строк в таблице переходов автомата D, для которых еще не вычислены переходы, применить шаг 2.

D

	0	*1*
{A, B}	{A,B}	{C}
{C}	{}	{A,C}
{ }		
{A,C}		

⇒

D

	0	*1*
{A, B}	{A,B}	{C}
{C}	{}	{A,C}
{ }	{ }	{ }
{A,C}	{A,B}	{A,C}

Алгоритм построения эквивалентного детерминированного автомата

5. Если строка таблицы переходов автомата D содержит состояние автомата N, то пометить соответствующее состояние автомата D как допускающее, иначе – как отвергающее.

D

	0	*1*
{A, B}	{A,B}	{C}
{C}	{}	{A,C}
{ }	{ }	{ }
{A,C}	{A,B}	{A,C}

⇒

D

	0	*1*	
{A, B}	{A,B}	{C}	1
{C}	{}	{A,C}	1
{ }	{ }	{ }	0
{A,C}	{A,B}	{A,C}	1

Алгоритм построения эквивалентного детерминированного автомата

5. Если строка таблицы переходов автомата D содержит состояние автомата N, то пометить соответствующее состояние автомата D как допускающее, иначе – как отвергающее.

D

	0	*1*	
{A, B}	{A,B}	{C}	1
{C}	{}	{A,C}	1
{ }	{ }	{ }	0
{A,C}	{A,B}	{A,C}	1

D

	0	*1*	
1	1	2	1
2	3	4	1
3	3	3	0
4	1	4	1

Алгоритм построения эквивалентного детерминированного автомата

D

	0	*1*	
1	1	2	1
2	3	4	1
3	3	3	0
4	1	4	1

- Замечание.
  Описанный алгоритм гарантирует, что полученный детерминированный автомат не содержит недостижимых состояний, но не гарантирует, что он будет минимальным. *(Почему?)*
- Упражнение.
  Как проверить эквивалентность состояний недетерминированного автомата?

Реализация конечного автомата

- Основные проблемы реализации автомата:
  - Представление входных символов
  - Представление состояний
  - Представление переходов
  - Идентификация слов

Представление входных символов

Сокращение исходного алфавита

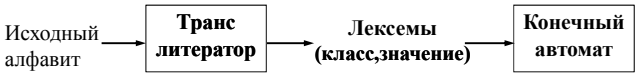


Таблица транслитерации

Символ	*A*	…	*Z*	*0*	…	*9*	…
Лексема	(БУКВА, A)	…	(БУКВА, Z)	(ЦИФРА, 0)	…	(ЦИФРА, 9)	…

Представление состояний

- *Явное представление* – переменная, в которой хранится текущее состояние.
- *Неявное представление* – для каждого состояния имеется отдельная подпрограмма; состояние определяется тем, какая именно подпрограмма выполняется.

Представление переходов

- *Метод вектора переходов*.
  Адреса подпрограмм обработки переходов хранятся в массиве, который индексируется входным алфавитом автомата.
- *Метод списка переходов*.
  Входной алфавит автомата делится на две группы: символы, для которых осуществляется *переход по неудаче* (которые обрабатываются подпрограммой обработки ошибок), и остальные.

Представление переходов

	1	*2*	*3*	*4*	*5*	*6*	*7*	⊣
A	**A1**			**B3**		**A6**		
B		**C2**	**C3**	**A4**		**B6**	**C1**	
C					**B5**			

Вектор переходов для состояния A

1	*2*	*3*	*4*	*5*	*6*	*7*	⊣
@A1	**@HET**	**@HET**	**@B3**	**@HET**	**@A6**	**@HET**	**@HET**

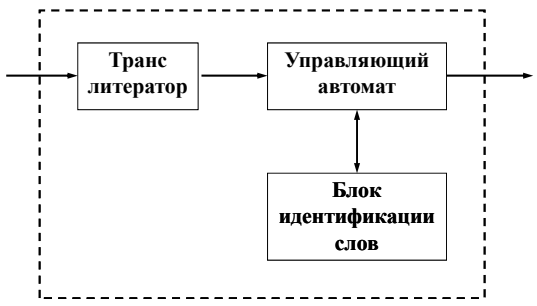
Список переходов для состояния A

1	*4*	*6*
@A1	**@B3**	**@A6**

Идентификация слов

- *Идентификация слов* – задано конечное множество слов (цепочек) во входном алфавите и входное слово. Нужно установить, какой элемент множества совпадает с входным словом, либо установить, что входное слово не принадлежит множеству.
- Идентификация ключевых слов языка программирования необходима при реализации лексического блока.

Схема структуры лексического блока



Методы идентификации слов

- Метод автомата
- Метод индексов
- Метод линейного списка
- Метод упорядоченного списка
- Метод расстановки (хеширование)

Метод автомата

- Множество состояний автомата, идентифицирующего слова, получается объединением следующих множеств:
  - множество всех префиксов множества слов (в том числе сами слова)
  - пустая цепочка ε

Пример идентифицирующего автомата

- Идентификация слов БАЗА, БАЗАР, БАР, БАС.

	А	*Б*	*З*	*Р*	*С*	⊣
ε		**Б**				
Б	**БА**					
БА			**БАЗ**	**БАР**	**БАС**	
БАР						**«БАР»**
БАС						**«БАС»**
БАЗ	**БАЗА**					
БАЗА				**БАЗАР**		**«БАЗА»**
БАЗАР						**«БАЗАР»**

- Пустые ячейки таблицы – вызов процедуры НЕТ.

Метод индексов

- Создается таблица, элементы которой – допустимые слова. Слово преобразуется в индекс, который является номером элемента таблицы.
- Пример: пусть в языке программирования
  <идентификатор>::=<буква><цифра>
  <буква>::=A | B | … | Z
  <цифра>::=0 | 1 | … | 9
  Тогда можно взять функцию индексирования
  Индекс(<идентификатор>)=*номер*(<буква>)+26*(*номер*(<цифра>)+1)
  Индекс(A)=1, Индекс(Z)=26, Индекс(A0)=27, Индекс(Z9)=286

Метод линейного списка

- Поиск входного слова в списке допустимых слов методом линейного поиска.
- Низкая эффективность метода при большом размере списка слов: в среднем требуется $(N+1)/2$ операций сравнения, где N – размер списка.
- Простота расширения списка допустимых слов.

Метод упорядоченного списка

- Поиск входного слова в *упорядоченном* списке допустимых слов методом бинарного поиска.
- Более высокая эффективность: в среднем требуется $1+\log_2 N$ операций сравнения.
- Трудность расширения списка допустимых слов.

Метод расстановки (хеширование)

- Индекс входного слова является номером элемента таблицы, в котором хранится список слов, индекс которых совпадает с индексом входного слова.
- Методы вычисления индекса:
  - По первым *k* символам слова, $1 \le k \le$ длина_слова
  - *Рандомизация* – индекс вычисляется с помощью двоичного кода входного слова (сцепление двоичных кодов номеров символов слова), например:
    - индекс – остаток от деления двоичного кода слова на простое число
    - индекс – средние двоичные разряды от квадрата двоичного кода
- Методы поиска значения индекса с возможной вставкой в таблицу – см.
  Кнут Д.Э. Искусство программирования, т. 3. Сортировка и поиск, 2-е изд. М.: Изд. дом "Вильямс", 2000. 832 с.

Заключение

- **Конечный автомат** – модель вычислительного устройства с фиксированным и конечным объемом памяти, которое читает и обрабатывает цепочку входных символов, принадлежащих некоторому конечному множеству.
- Основные разновидности конечного автомата – **конечный распознаватель** и **обрабатывающий автомат**.
- Для построения конечного распознавателя можно использовать **метод разметки символов**.

Заключение

- Для любого конечного автомата существует единственный **минимальный автомат**. Приведение автомата к минимальному называется **редукцией**.
- Для редукции конечного автомата необходимо найти **эквивалентные состояния** и заменить их на одно и отбросить **недостижимые состояния**.
- Поиск эквивалентных состояний можно проводить с помощью построения **таблиц эквивалентности** или **методом разбиения**.

Заключение

- **Недетерминированный конечный автомат** – конечный автомат, у которого значением функции перехода является *множество* состояний.
- Для любого недетерминированного конечного автомата можно построить **эквивалентный** ему детерминированный конечный автомат.

Заключение

- Основные вопросы, возникающие при реализации конечных автоматов – **представление входных символов** и **состояний**, **организация переходов** и **идентификация слов**.
- Реализация лексического блока компилятора предполагает введение в его структуру **транслитератора** и **блока идентификации слов**.