УДК 519.685.3

АНАЛИЗ СИНТАКСИЧЕСКИХ ДИАГРАММ И СИНТЕЗ ПРОГРАММ-РАСПОЗНАВАТЕЛЕЙ ЛИНЕЙНОЙ СЛОЖНОСТИ

Ю. Д. РЯЗАНОВ
М. Н. СЕВАЛЬНЕВА

Белгородский государственный технологический университет им. В. Г. Шухова

e-mail: Ryazanov.iurij@yandex.ru

Рассматриваются вопросы использования синтаксических диаграмм для автоматизации проектирования трансляторов. Определен класс детерминированных синтаксических диаграмм, разработан алгоритм анализа табличного представления синтаксической диаграммы с целью определения ее принадлежности классу детерминированных диаграмм и алгоритм синтеза программы-распознавателя линейной сложности.

Ключевые слова: транслятор, детерминированная синтаксическая диаграмма, множество выбора, программа-распознаватель.

Введение. Одним из удобных способов описания формальных языков является синтаксическая диаграмма (СД). Этот графический, оперирующий образами способ ориентирован на человека и в основном применялся в руководствах по языкам программирования [1]. Сейчас известны примеры использования СД при инженерном, ручном проектировании трансляторов [2—6]. В системах автоматизированного построения трансляторов [7—10] СД не применяются. Как правило, спецификация транслятора в этих системах представляется в нотации РБНФ, выполняется классическое проектирование транслятора [11] и его реализация с использованием алгоритмов линейной сложности, если это позволяет исходная РБНФ, либо с использованием универсальных алгоритмов более высокой сложности, таких как GLR, GLL или алгоритмов с возвратами.

Здесь дается формальное определение СД, позволяющее использовать табличное представление СД, удобное для автоматической обработки, определяется класс детерминированных СД, которые можно преобразовать в программы линейной сложности, предлагаются алгоритмы анализа таблиц СД с целью определения принадлежности СД классу детерминированных СД и алгоритм преобразования таблиц детерминированных СД в программу-распознаватель линейной сложности.

Табличный способ представления СД и предложенные алгоритмы ориентированы на их использование в системах автоматизированного проектирования трансляторов.

Определение синтаксической диаграммы. Синтаксическую диаграмму будем задавать четверкой $D = (T, N, S, G)$, где T — конечное множество терминалов; N — конечное множество нетерминалов; $S \in N$ — начальный нетерминал; $G = (V, E)$ — ориентированный граф, где

$V = V_T \cup V_N \cup V_u \cup V_{вход} \cup V_{выход}$, где

$V_{вход}$ — множество точек входа, $|V_{вход}| = |N|$;

V_T — множество терминальных вершин;

V_N — множество нетерминальных вершин;

V_u — конечное множество узлов;

$V_{выход}$ — множество точек выхода, $|V_{выход}| = |N|$;

$E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$, где

$E_1 \subseteq \{(a, b) \mid a \in V_{вход}, b \in V_u\}$ — множество входных дуг;

$E_2 \subseteq \{(a, b) \mid a \in V_u, b \in V_{выход}\}$ — множество выходных дуг;

$E_3 \subseteq \{(a, b) \mid a \in V_u, b \in V_T \cup V_N\}$ — множество дуг, выходящих из узлов;

$E_4 \subseteq \{(a, b) \mid a \in V_T \cup V_N, b \in V_u\}$ — множество дуг, входящих в узлы;

$E_5 \subseteq \{(a, b) \mid a \in V_u, b \in V_u\}$ — множество ε-дуг, соединяющих узлы.

Каждому нетерминалу соответствует связная компонента графа. Компонента именуется соответствующим нетерминалом, имеет только одну точку входа и одну точку выхода и конечное множество вершин других типов. Точки входа и выхода на диаграмме компоненты не изображаются. Нетерминальная вершина изображается прямоугольником, в который вписан терминальный символ. Узел изображается на диаграмме жирной точкой. В точку входа не входит ни одна дуга и выходит конечное множество дуг (входные дуги компоненты). Узлы, в которые входят входные дуги, называются начальными. Из точки выхода не выходит ни одна дуга и входит конечное множество дуг (выходные дуги компоненты). Узлы, из которых выходят выходные дуги, называются заключительными. Каждая дуга, за исключением входных и выходных дуг, может выходить из узла и входить в терминальную или нетерминальную вершину или другой узел, либо выходить из терминальной или нетерминальной вершины и входить в узел. В каждую терминальную и нетерминальную вершину входит только одна дуга и выходит только одна дуга. На количество дуг, входящих в узлы и выходящих из них, ограничений нет. На рис. 1 приведен пример синтаксической диаграммы.

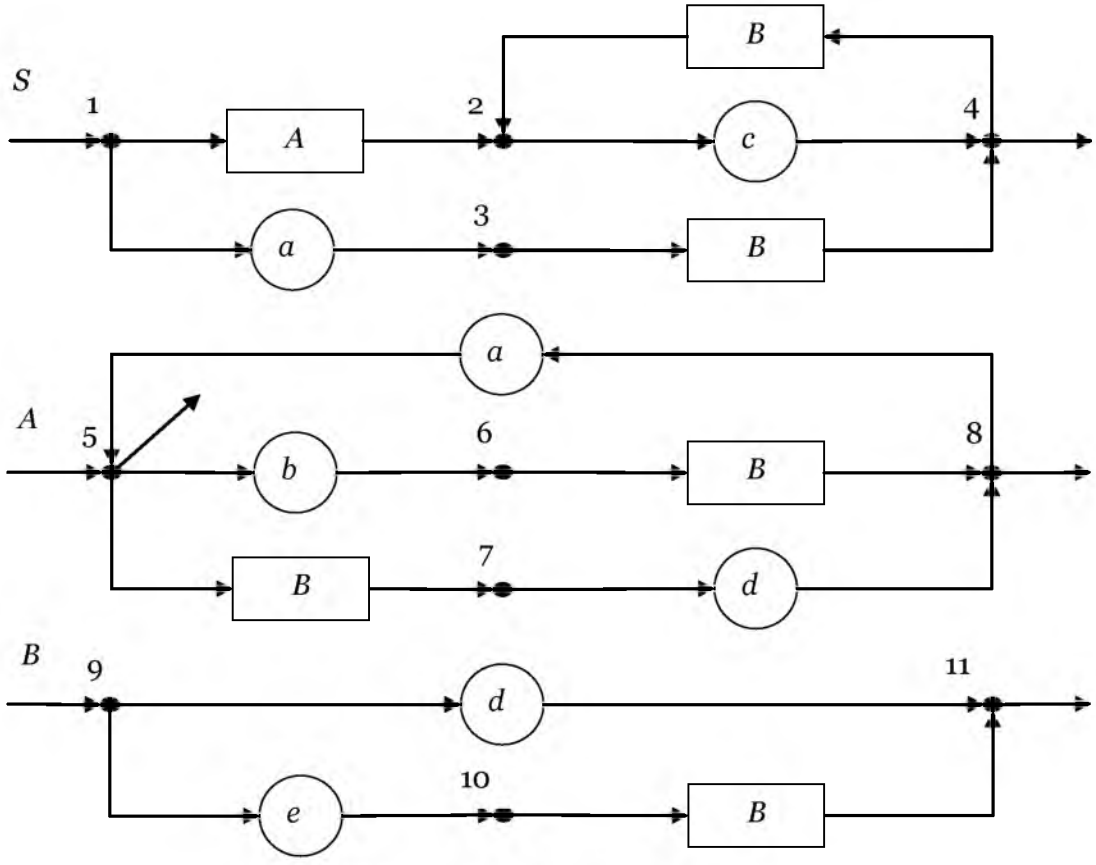

Рис. 1. Синтаксическая диаграмма

Таблицы синтаксических диаграмм. Синтаксическую диаграмму D можно представить множеством таблиц T, по одной для каждой компоненты.

Определим функцию $R : U \to \{1, 2,..., |V_u|\}$ разметки узлов, которая каждому узлу синтаксической диаграммы ставит во взаимнооднозначное соответствие элемент из множества $\{1, 2,..., |V_u|\}$.

Компоненту A синтаксической диаграммы будем представлять одноименной таблицей A, содержащей m строк и n столбцов, где n — количество узлов в компоненте A, $m = |T|+|N|+1$. Столбцы таблицы соответствуют узлам и отмечаются метками в соответствии с функцией R. Столбцы, соответствующие начальным узлам, отмечаются

стрелочками. Столбцы, соответствующие заключительным узлам, отмечаются символом 1. Строки отмечаются терминалами, нетерминалами и символом ε. Каждый элемент таблицы содержит некоторое, возможно пустое, множество меток.

Пустая таблица A заполняется по следующим правилам. Если в диаграмме компоненты из узла с меткой i существует путь в узел с меткой j, проходящий только через одну терминальную вершину с терминалом $a \in T$, то в элемент таблицы $A_{a,i}$ добавляем метку j. Если из узла с меткой i существует путь в узел с меткой j, проходящий только через одну нетерминальную вершину с нетерминалом $B \in N$, то в элемент таблицы $A_{B,i}$ добавляем метку j. Если существует ε-дуга из узла i в узел j, то в элемент таблицы $A_{\varepsilon,i}$ добавляем метку j. Синтаксическая диаграмма (см. рис. 1) в табличной форме представлена в табл. 1.

Таблица 1

Табличное представление синтаксической диаграммы

S	↓			1
	1	**2**	**3**	**4**
a	3			
b				
c		4		
d				
e				
S				
A	2			
B			4	2

A	↓ 1			1
	5	**6**	**7**	**8**
a				5
b	6			
c				
d			8	
e				
S				
A				
B	7	8		

B	↓		1
	9	**10**	**11**
a			
b			
c			
d	11		
e	10		
S			
A			
B		11	

Вывод в синтаксической диаграмме. Пусть в компоненте A синтаксической диаграммы существует путь l от точки входа до точки выхода. Если начать движение по этому пути и, проходя через терминальную или нетерминальную вершину, переписывать символ из вершины в некоторую изначально пустую цепочку α, то по окончании движения, когда придем в точку выхода, будет сформирована цепочка α, соответствующая пути l. Если путь l от точки входа до точки выхода не проходит ни через одну терминальную или нетерминальную вершину, то цепочка α пустая.

Пусть L_A — множество всех путей от точки входа до точки выхода в компоненте A. Определим функцию $F : L_A \to \alpha_A$, которая каждому пути $l \in L_A$ ставит в соответствие цепочку α по описанному выше правилу.

Выводимые в СД цепочки определим следующим образом:

1) цепочка S — выводимая, где S — начальный нетерминал;

2) если цепочка $\gamma A \beta$ выводимая, где γ и β — цепочки, состоящие из терминалов и нетерминалов, возможно пустые, а A — нетерминал, и некоторая цепочка $\alpha \in \alpha_A$, то цепочка $\gamma\alpha\beta$ выводимая. Цепочка $\gamma\alpha\beta$ получается из цепочки $\gamma A \beta$ путем замены нетерминала A на цепочку $\alpha \in \alpha_A$.

Будем говорить, что цепочка $\gamma\alpha\beta$ непосредственно выводится из $\gamma A \beta$ и записывать $\gamma A \beta \Rightarrow \gamma\alpha\beta$. Выводимая в СД цепочка, содержащая хотя бы один нетерминал, называется промежуточной, а цепочка, не содержащая нетерминалов — терминальной.

Выводом в СД называется последовательность замен нетерминалов A в промежуточных цепочках на цепочки $\alpha \in \alpha_A$. Вывод заканчивается получением терминальной цепочки. Терминальная цепочка принадлежит языку, заданному синтаксической диаграммой.

Если на каждом шаге вывода заменяется самый левый нетерминал в промежуточной цепочке, то вывод называется левым.

Псевдодетерминированные синтаксические диаграммы.СД назовем псевдодетерминированной, если все ее компоненты псевдодетерминированные.

Компоненту СД назовем псевдодетерминированной, если:

1) содержит только один начальный узел;

2) не имеет ε-дуг;

3) все ее узлы — псевдодетерминированные.

Узел u компоненты A синтаксической диаграммы назовем псевдодетерминированным, если любые две дуги, выходящие из узла u, входят в вершины, содержащие различные символы.

Отличительной особенностью таблицы A псевдодетерминированной компоненты является следующее:

1) имеет только один начальный узел;

2) все элементы в строке ε — пустые;

3) в остальных строках таблицы элементы либо пустые, либо одноэлементные множества.

Любую СД можно преобразовать в эквивалентную ей псевдодетерминированную диаграмму [12].

Детерминированные синтаксические диаграммы.СД назовем детерминированной, если все ее компоненты — детерминированные.

Компоненту СД назовем детерминированной, если она псевдодетерминированная и каждый ее узел детерминированный.

Узел u назовем детерминированным, если множества выбора (определяется ниже) любых двух дуг, выходящих из узла u, не пересекаются. Отметим, что дуги, выходящие из заключительного узла псевдодетерминированной компоненты СД, могут идти в терминальную, нетерминальную вершину или в точку выхода.

Процесс левого вывода терминальной цепочки в СД можно представить как «движение» по дугам от точки входа начальной компоненты к ее точке выхода. При этом если дуга идет в терминальную вершину, то вписанный в нее символ добавляем в терминальную цепочку, если дуга идет в нетерминальную вершину, то переходим в соответствующую компоненту и движемся по ней аналогичным образом до точки выхода, после чего возвращаемся в предыдущую компоненту и продолжаем движение. После прохождения выходной дуги начальной компоненты в терминальную цепочку добавляем концевой маркер и вывод заканчивается.

Символ x, который может быть добавлен в терминальную цепочку непосредственно после прохождения дуги e, принадлежит множеству выбора дуги e ($x \in$ ВЫБОР(e)).

Для нахождения множества выбора дуг СД определим множества первых и следующих для нетерминалов.

Множество первых для нетерминала X (ПЕРВ(X)) содержит в себе все те терминалы, с которых начинаются терминальные цепочки, выводимые из нетерминала X. Если из нетерминала X выводится пустая цепочка, то символ ε принадлежит множеству ПЕРВ(X).

Алгоритм формирования множеств ПЕРВ *для каждого нетерминала* СД.

Вход: множество таблиц СД.

Выход: ПЕРВ(X) — множество первых для каждого нетерминала X СД.

1. Для всех нетерминалов X выполнить ПЕРВ(X) := $\varnothing$.

2. Для всех нетерминалов X выполнить процедуру ПервУзла(u), где u — начальный узел компоненты X.

3. Повторять п.2, пока множества ПЕРВ(X) изменяются.

4. Конец алгоритма.

Алгоритм процедуры ПервУзла(u).

Вход: u — узел обрабатываемой компоненты X.

Глобальные параметры:

X — таблица обрабатываемой компоненты X;

ПЕРВ — массив множеств ПЕРВ каждого нетерминала.

1. Если u — заключительный узел, то ПЕРВ(X) := ПЕРВ(X)$\cup\{\varepsilon\}$.
2. Для всех дуг (u, x), выходящих из узла u выполнить:
   2.1. Если x — терминал, то ПЕРВ(X) := ПЕРВ(X)$\cup\{x\}$.
   2.2. Если x — нетерминал, то ПЕРВ (X) := ПЕРВ(X)$\cup$(ПЕРВ(x)$\setminus\{\varepsilon\}$) и если $\varepsilon\in$ПЕРВ(x), то выполнить процедуру ПервУзла($X_{x,u}$), т. е. процедуру ПервУзла для узла, в который идет дуга из вершины с нетерминалом x.
3. Конец алгоритма.

Множество следующих для нетерминала X (СЛЕД(X)) включает в себя те терминалы, которые могут появиться в какой-либо промежуточной цепочке вывода непосредственно после нетерминала X. Так как любая цепочка заканчивается концевым маркером, а вывод начинается с цепочки, содержащей только начальный нетерминал, то множество следующих для начального нетерминала содержит концевой маркер (⊣).

Алгоритм формирования множеств СЛЕД *для каждого нетерминала* СД.

Вход: множество таблиц СД.

Выход: СЛЕД(X) — множество следующих для каждого нетерминала X СД.

1. Для начального нетерминала S СЛЕД(S): = {⊣}, для всех остальных нетерминалов X выполнить СЛЕД(X) := $\varnothing$.
2. Для всех нетерминалов X выполнить:
   для каждого узла u, в который идет дуга из вершины с нетерминалом X (определяется непустыми элементами в строке X таблиц СД), выполнить процедуру СледУзла(u).
3. Повторять п.2, пока множества СЛЕД(X) изменяются.
4. Конец алгоритма.

Алгоритм процедуры СледУзла(u).

Вход: u — обрабатываемый узел.

Глобальные параметры:

A — таблица компоненты A, которой принадлежит узел u;

X — нетерминал, для которого формируется множество СЛЕД;

СЛЕД — массив множеств СЛЕД каждого нетерминала.

1. Если u—заключительный узел компоненты A, то СЛЕД(X):=СЛЕД(X)$\cup$СЛЕД (A).
2. Для всех дуг (u, x), выходящих из узла u в вершину с символом x выполнить:
   2.1. Если x — терминал, то СЛЕД(X) := СЛЕД(X)$\cup\{x\}$.
   2.2. Если x — нетерминал, то СЛЕД (X) := СЛЕД(X)$\cup$(ПЕРВ(x)$\setminus\{\varepsilon\}$) и если $\varepsilon\in$ПЕРВ(x), то выполнить процедуру СледУзла($A_{x,u}$), т. е. процедуру СледУзла для узла, в который идет дуга из вершины с нетерминалом x.
3. Конец алгоритма.

Применяя эти алгоритмы к СД, представленной на рис. 1, получим множества ПЕРВ и СЛЕД для каждой компоненты, представленные в табл. 2.

Таблица 2

Множества ПЕРВ и СЛЕД

	ПЕРВ	СЛЕД
S	$\{a, b, c, d, e\}$	{⊣}
A	$\{b, d, e\}$	$\{c\}$
B	$\{d, e\}$	$\{a, c, d,$ ⊣$\}$

Множество выбора для дуги (u, x) (ВЫБОР(u, x)) компоненты A определяется следующим образом:

1) если x — терминал, то ВЫБОР(u, x) := $\{x\}$;
2) если x — нетерминал, то ВЫБОР(u, x) := ВЫБОР(u, x)$\cup$(ПЕРВ(x)$\setminus\{\varepsilon\}$)и если $\varepsilon\in$ПЕРВ (x), то выполнить процедуру ВыборУзла ($A_{x,u}$);
3) если u — заключительный узел и x = выход (дуга (u, x) —выходная), то ВЫБОР (u, x) := СЛЕД(A).

Алгоритм процедуры ВыборУзла(y).

Вход: y — обрабатываемый узел.
Глобальные параметры:
A — таблица компоненты A, которой принадлежит узел u;
ВЫБОР(u, x) — множество выбора для дуги (u, x).
1. Если y — заключительный узел компоненты A, то
ВЫБОР(u, x) := ВЫБОР(u, x)∪СЛЕД(A).
2. Для всех дуг (y, z), выходящих из узла y в вершину с символом z выполнить:
2.1. Если z — терминал, то СЛЕД(X) := СЛЕД(X)∪{z}.
2.2. Если z — нетерминал, то ВЫБОР(u, x) := ВЫБОР(u, x)∪(ПЕРВ(z)\{ε}) и если ε∈ПЕРВ(z), то выполнить процедуру ВыборУзла($A_{z,y}$), т. е. процедуру ВыборУзла для узла, в который идет дуга из вершины с нетерминалом z.
3. Конец алгоритма.

Результат применения алгоритма нахождения множеств выбора для дуг компонент СД (см. рис. 1), представлен в табл. 3.

Для определения детерминированности узла необходимо просмотреть соответствующий столбец в таблице и убедиться, что множества, записанные в любой паре ячеек этого столбца, не пересекаются. Анализируя столбцы таблиц каждой компоненты, видим, что все узлы детерминированные. Таким образом, синтаксическая диаграмма, представленная на рис. 1, принадлежит классу детерминированных диаграмм.

Таблица 3

Множества выбора для дуг синтаксической диаграммы

S	↓			1
	1	**2**	**3**	**4**
a	*a*			
b				
c		*c*		
d				
e				
S				
A	*b,c,d,e*			
B			*d,e*	*d,e*
выход				-┤

A	↓1			1
	5	**6**	**7**	**8**
a				*a*
b	*b*			
c				
d			*d*	
e				
S				
A				
B	*d,e*	*d,e*		
выход	*3*			*3*

B	↓		1
	9	**10**	**11**
a			
b			
c			
d	*d*		
e	*e*		
S			
A			
B		*d,e*	
выход			*a,c,d,-*┤

Синтез программ-распознавателей по таблицам детерминированных синтаксических диаграмм. Не ориентируясь на конкретный язык программирования, будем представлять программу-распознаватель на псевдокоде. Дадим краткое неформальное описание элементов и команд псевдокода:

1) *читать(x)* — чтение очередного символа входной цепочки в переменную x;
2) *Программа* и *Конец программы* — начало и конец основной программы;
3) *Подпрограмма A* и *Конец A* — начало и конец подпрограммы, соответствующей компоненте A;
4) *Вызов A* — обращение к подпрограмме, соответствующей компоненте A;
5) *выход* — выход из подпрограммы;
6) 1, 2, 3 ... — метки, соответствующие узлам СД;
7) операции отношения: $x=t$ и $x\in\{a, b, c\}$, где x — переменная, хранящая обрабатываемый символ входной цепочки; t — терминал или концевой маркер (┤); {a, b, c} — множество терминалов;
8) *если* <отношение> *то* <Оператор1> *иначе* <Оператор2> — условный оператор, где <Оператор1> может представлять собой один из четырех вариантов:
— *читать(x), переход на* <метка>;

— *Вызов A, переход на* <метка>;
— *выход*;
— *Допустить*;
<Оператор2> — условный оператор или *Отвергнуть, конец.*

Программа-распознаватель представляет собой множество подпрограмм, соответствующих компонентам СД, и основную программу, в которой читается первый символ входной цепочки и вызывается подпрограмма, соответствующая начальному нетерминалу. Если после выхода из этой подпрограммы анализируемый символ входной цепочки представляет собой концевой маркер, то цепочка допускается, иначе — отвергается.

Текст основной программы-распознавателя для распознавания цепочек языка, заданного СД (см. рис. 1 и табл. 1) представлен ниже на псевдокоде:

Программа
читать (x);
Вызов S;
если x = ┤ *то Допустить иначе Отвергнуть*
Конец программы.

Подпрограмма, соответствующая компоненте *A* СД, представляет собой последовательность фрагментов кода, описывающих узлы компоненты. Каждому узлу соответствует фрагмент кода, который отмечен одноименной меткой и представляет собой последовательность описаний дуг, выходящих из узла. Дуга описывается условным оператором, в котором определяется принадлежность обрабатываемого символа множеству выбора дуги. Если символ принадлежит множеству выбора и дуга идет в терминальную вершину, то читается следующий символ входной цепочки и выполняется переход на метку, соответствующую узлу, в который идет дуга из терминальной вершины. Если символ принадлежит множеству выбора и дуга идет в нетерминальную вершину, то вызывается подпрограмма, соответствующая нетерминалу, записанному в вершине, и, после выхода из нее, выполняется переход на метку, соответствующую узлу, в который идет дуга из нетерминальной вершины. Если символ не принадлежит множеству выбора и дуга не последняя, то описывается следующая дуга, выходящая из рассматриваемого узла, а если дуга последняя, то цепочка отвергается. Выходная дуга заключительного узла, идущая в точку выхода, описывается последней. Если символ принадлежит множеству выбора, то происходит выход из подпрограммы, иначе цепочка отвергается.

Текст подпрограмм, соответствующих компонентам СД (см. рис. 1, табл. 1 и 3) представлен ниже на псевдокоде:

Подпрограмма S
1: *если x = a то читать (x), переход на* 3 *иначе*
если x ∈ {b, c, d, e} то Вызов A, переход на 2 *иначе Отвергнуть, конец.*
2: *если x = c то читать (x), переход на* 4 *иначе Отвергнуть, конец.*
3: *если x ∈ {d, e} то Вызов B, переход на* 4 *иначе Отвергнуть, конец.*
4: *если x ∈ {d, e} то Вызов B, переход на* 2 *иначе*
если x ∈ { ┤ *} то выход иначе Отвергнуть, конец.*
Конец S
Подпрограмма A
5: *если x = b то читать (x), переход на* 6 *иначе*
если x ∈ {d, e} то Вызов B, переход на 7 *иначе*
если x ∈ {c} то выход иначе Отвергнуть, конец.
6: если x ∈ {d, e} то Вызов B, переход на 8 *иначе Отвергнуть, конец.*
7: *если x = d то читать (x), переход на* 8 *иначе Отвергнуть, конец.*
8: *если x = a то читать (x), переход на* 5 *иначе*
если x ∈ {c} то выход иначе Отвергнуть, конец.
Конец A
Подпрограмма B
9: *если x = d то читать (x), переход на* 11 *иначе*

если x = e то читать (x), переход на 10 *иначе Отвергнуть, конец.*

10: *если x ∈ {d, e} то Вызов B, переход на* 11 *иначе Отвергнуть, конец.*

11: *если x ∈ {a, c, d, ┤} то выход иначе Отвергнуть, конец.*

Конец B

Текст основной программы-распознавателя не зависит от СД, поэтому алгоритм его формирования тривиален.

Алгоритм синтеза подпрограмм, соответствующих компонентам СД следующий: для всех компонент СД сформировать подпрограмму, используя процедуру СинтезПодпрограммы.

Алгоритм процедуры СинтезПодпрограммы(*A*, *MV*).

Вход: *A* — таблица компоненты *A*;

MV — таблица, хранящая множества выбора дуг, выходящих из узлов компоненты *A*.

Выход: текст подпрограммы, соответствующий компоненте *A*.

1. Выдать «*Подпрограмма A*»
2. Для всех узлов *u* компоненты *A* (столбцов таблицы *A*) выполнить:
   2.1. Выдать метку, соответствующую узлу *u*.
   2.2. Для всех строк-терминалов *t* выполнить:
   если $A_{t,u} \neq \varnothing$, то выдать «*если x = t то читать (x), переход на m иначе*», где $m = A_{t,u}$.
   2.3. Для всех строк-нетерминалов *Y* выполнить:
   если $A_{Y,u} \neq \varnothing$, то выдать «*если* $x \in MV_{Y,u}$ *то Вызов Y , переход на m иначе*», где $m = A_{Y,u}$.
   2.4. Если $A_{\text{выход},u} \neq \varnothing$, то выдать «*если* $x \in MV_{\text{выход},u}$ *то выход иначе*».
   2.5. Выдать «*Отвергнуть, конец.*».
3. Выдать «*Конец A*».

Заключение. В статье определен класс детерминированных синтаксических диаграмм, разработан алгоритм анализа табличного представления синтаксической диаграммы с целью определения ее принадлежности классу детерминированных диаграмм и алгоритм синтеза программы-распознавателя линейной сложности. Эти алгоритмы реализованы в разрабатываемой системе автоматизированного построения трансляторов на основе синтаксических диаграмм.

Литература

1. Йенсен К., Вирт Н. Паскаль. Руководство для пользователя и описание языка. — М: Финансы и статистика, 1982. — 151 с.

2. Легалов А. И. Основы разработки трансляторов. URL: http://www.softcraft.ru/translat/lect/content.shtml

3. Легалов А. И., Швец Д. А., Легалов И. А. Формальные языки и трансляторы. — Красноярск: Сибирский федеральный университет, 2007. — 213 с.

4. Свердлов С. З. Введение в методы трансляции. — Вологда: Издательство «Русь», 1994. — 80 с.

5. Свердлов С. З. Языки программирования и методы трансляции.— СПб.: Питер, 2007. — 638 с.

6. Карпов Ю. Г. Теория и технология программирования. Основы построения трансляторов. — СПб.: БХВ-Петербург, 2005. — 272 с.

7. ANTLR. URL: http://www.antlr.org/

8. ASF+SDF. URL: http://www.cwi.nl/projects/MetaEnv/

9. Bison. URL: http://www.gnu.org/software/bison/

10. Coco/R. URL: http://www.ssw.uni-linz.ac.at/Research/Projects/Coco/

11. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. — М.: Мир, 1978. Т. 1, 612 с. — т. 2, 487 с.

12. Рязанов Ю. Д., Севальнева М. Н. Псевдодетерминированные синтаксические диаграммы // Прикладная математика, управление и информатика: сборник трудов Междунар. молодеж. конф., Белгород, 3 — 5 октября 2012 г.: в 2 т. — Белгород: ИД «Белгород», 2012. Т. 2. С. 546 — 553.

THE ANALYSIS OF SYNTAX DIAGRAMS AND AUTOMATIC GENERATION OF LINEAR-TIME PROGRAMS-RECOGNIZER

Y. D. RYAZANOV
M. N. SEVAI'NEVA

Belgorod Shukhov State Technology University

e-mail: Ryazanov.iurij@yandex.ru

There are considered the problems of applying syntax diagrams to automatize the translators designing. There is determined the class of determined syntax diagrams, developed an algorithm analysis tabular representation syntax diagram to define its belongs to the class of deterministic diagrams and algorithm of automatic generation of linear-time program-recognizer.

Keywords: translator, determined syntax diagrams, set of a choice, the program-recognizer.