Ф. ЛЬЮИС,
Д. РОЗЕНКРАНЦ,
Р. СТИРНЗ

Теоретические основы проектирования компиляторов

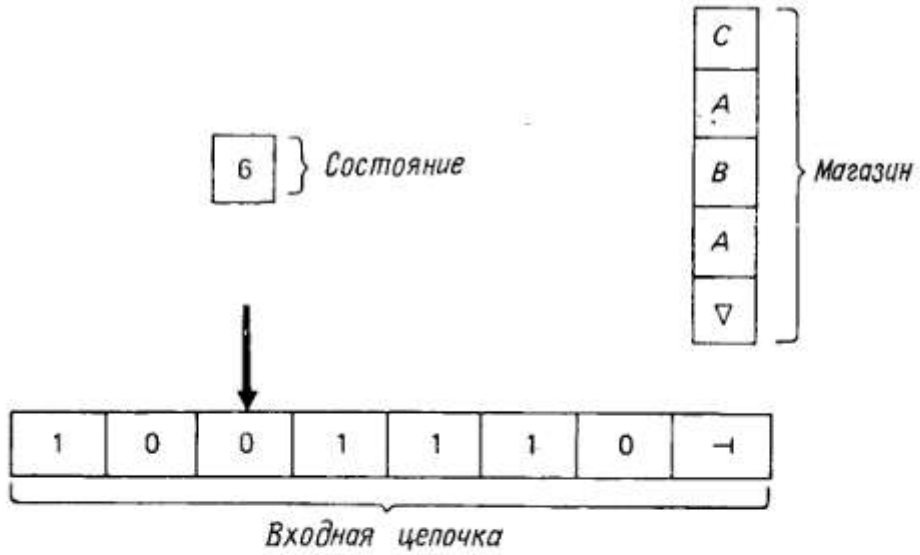

Рис. 5.2.

Одной из моделей автомата, в которых используется магазинный принцип организации памяти, является *автомат с магазинной памятью* (или сокращенно *МП-автомат*). В нем очень просто комбинируется память конечного автомата и магазинная память. МП-автомат может находиться в одном из конечного числа состояний и имеет магазин, куда он может помещать и откуда может извлекать информацию.

Как и в случае конечного автомата, обработка входной цепочки осуществляется за ряд мелких шагов. На каждом шаге действия автомата конфигурация его памяти может измениться за счет перехода в новое состояние, а также вталкивания символа в магазин или выталкивания из него. Однако в отличие от конечного автомата. МП-автомат может обрабатывать один входной символ в течение нескольких шагов. На каждом шаге управляющее устройство автомата решает, пора ли закончить обработку текущего входного символа и получить, если это так, новый входной символ или продолжить обработку текущего символа на следующем шаге. На рис. 5.2 изображена одна из конфигураций, которая может возник-

нуть при обработке некоторым гипотетическим МП-автоматом входной цепочки 100110. Для большей наглядности входная цепочка изображена записанной в ячейках файла или ленты с указателем на входной символ, подвергающийся в данный момент обработке.

Каждый шаг процесса обработки задается множеством правил, использующих информацию трех видов:

1) состояние,
2) верхний символ магазина,
3) текущий входной символ.

Это множество правил называется управляющим устройством или механизмом управления. На рис. 5.2 информация, поступающая в управляющее устройство, такова: состояние 6, верхний символ магазина C, текущий входной символ 0.

В зависимости от получаемой информации, управляющее устройство выбирает либо выход из процесса (т. е. прекращает обработку), либо переход в новое состояние. Переход состоит из трех операций: над магазином, над состоянием и над входом.
Возможные операции таковы:

Операции над магазином

1. Втолкнуть в магазин определенный магазинный символ.
2. Вытолкнуть верхний символ магазина.
3. Оставить магазин без изменений.

Операция над состоянием

1. Перейти в заданное новое состояние.

Операции над входом

1. Перейти к следующему входному символу и сделать его текущим входным символом.
2. Оставить данный входной символ текущим, иначе говоря, держать его до следующего шага.

Обработку входной цепочки МП-автомат начинает в некотором выделенном состоянии при определенном содержимом магазина, а текущим входным символом является первый символ входной цепочки. Затем автомат выполняет операции, задаваемые его управляющим устройством. Если происходит выход из процесса, обработка прекращается. Если происходит переход, то он дает новый верхний магазинный символ, новый текущий символ, автомат переходит в новое состояние и управляющее устройство определяет новое действие, которое нужно произвести.

Чтобы управляющие правила имели смысл, автомат не должен требовать следующего входного символа, если текущим символом является концевой маркер, и не должен выталкивать символ из магазина, если это маркер дна. Поскольку маркер дна может находиться исключительно на дне магазина, автомат не должен также вталкивать его в магазин.

Теперь подытожим, как задается МП-автомат [1]). Он определяется следующими пятью объектами:

1) конечным множеством *входных символов*, в которое входит и концевой маркер;
2) конечным множеством *магазинных символов*, включающим маркер дна;
3) конечным множеством *состояний*, включающим начальное состояние;
4) *управляющим устройством*, которое каждой комбинации входного символа, магазинного символа и состояния ставит в соответствие выход или переход. Переход в отличие от выхода заключается в выполнении операций над магазином, состоянием и входом, как было описано выше. Операции, которые запрашивали бы входной символ после концевого маркера или выталкивали из магазина, а также вталкивали в него маркер дна, исключаются;
5) *начальным содержимым магазина*, которое представляет собой (при условии, что верхний символ считается расположенным справа) маркер дна, за которым следует (возможно, пустая) цепочка других магазинных символов.

МП-автомат называется *МП-распознавателем*, если у него два выхода — ДОПУСТИТЬ и ОТВЕРГНУТЬ. Говорят, что цепочка символов входного алфавита (исключая концевой маркер) *допускается* распознавателем, если под действием этой цепочки с концевым маркером автомат, начавший работу в своем начальном состоянии и с начальным содержимым магазина, делает ряд переходов, приводящих к выходу ДОПУСТИТЬ. В противном случае цепочка *отвергается*.

При описании переходов МП-автомата будем обозначать действия автомата словами ВЫТОЛКНУТЬ (или для краткости ВЫТОЛК), ВТОЛКНУТЬ (или ВТОЛК), СОСТОЯНИЕ, СДВИГ и ДЕРЖАТЬ, причем:

ВЫТОЛКНУТЬ означает вытолкнуть верхний символ магазина,

ВТОЛКНУТЬ (A), где A — магазинный символ, означает вытолкнуть символ A в магазин,

---

[1]) Заметим, что описываемая здесь и используемая до конца этой книги модель в теории автоматов называется детерминированным МП-автоматом, тогда как МП-автомат в общем случае может быть недетерминированным (ср. с недетерминированным конечным автоматом в разд. 2.12).— *Прим. ред.*

СОСТОЯНИЕ (s), где s — состояние, означает, что следующим состоянием становится s,

СДВИГ означает, что текущим входным символом становится следующий входной символ. В некоторых реализациях это может означать сдвиг указателя на входе,

ДЕРЖАТЬ означает, что текущий входной символ надо держать до следующего шага, т. е. оставить его текущим (в некоторых реализациях — оставить указатель на прежнем месте).

Когда нам нужно определить переход, который оставляет содержимое магазина неизменным, это выражается в том, что мы опускаем слова ВЫТОЛКНУТЬ и ВТОЛКНУТЬ. Хотя ДЕРЖАТЬ по существу означает, что СДВИГ отсутствует, мы всегда будем записывать операции над входом в явном виде, чтобы читателю было понятнее, что происходит. Если автомат содержит в точности одно состояние, мы будем опускать слово СОСТОЯНИЕ.

Сейчас мы опишем, как применить МП-распознаватель к проблеме скобок. Каждый раз, когда встречается левая скобка, в магазине будет вталкиваться символ A. Когда будет обнаружена соответствующая правая скобка, символ A будет выталкиваться из магазина. Цепочка отвергается, если на входе остаются правые скобки, а магазин пуст (т. е. во входной цепочке есть лишние правые скобки) или если цепочка прочитана до конца, а в магазине остаются символы A (т. е. входная цепочка содержит лишние левые скобки). Цепочка допускается, если к моменту прочтения входной цепочки до конца магазин опустошается. Полное определение таково:

1. Входное множество $\{(,), \dashv\}$.
2. Множество магазинных символов $\{A, \nabla\}$.
3. Множество состояний $\{s\}$, где s — начальное состояние.
4. Переходы:

$(, A, s =$ ВТОЛКНУТЬ (A), СОСТОЯНИЕ (s), СДВИГ
$(, \nabla, s =$ ВТОЛКНУТЬ (A), СОСТОЯНИЕ (s), СДВИГ
$), A, s =$ ВЫТОЛКНУТЬ, СОСТОЯНИЕ (s), СДВИГ
$), \nabla, s =$ ДЕРЖАТЬ
$\dashv, A, s =$ ДЕРЖАТЬ
$\dashv, \nabla, s =$ ДОПУСТИТЬ

Здесь комбинации входного символа, магазинного символа и состояния расположены слева от знака равенства, а переходы — справа от него.

5. Начальное содержимое магазина ∇.

Чтобы продемонстрировать работу автомата, мы изобразили на рис. 5.3, как он обрабатывает цепочку

(( ) ( ))

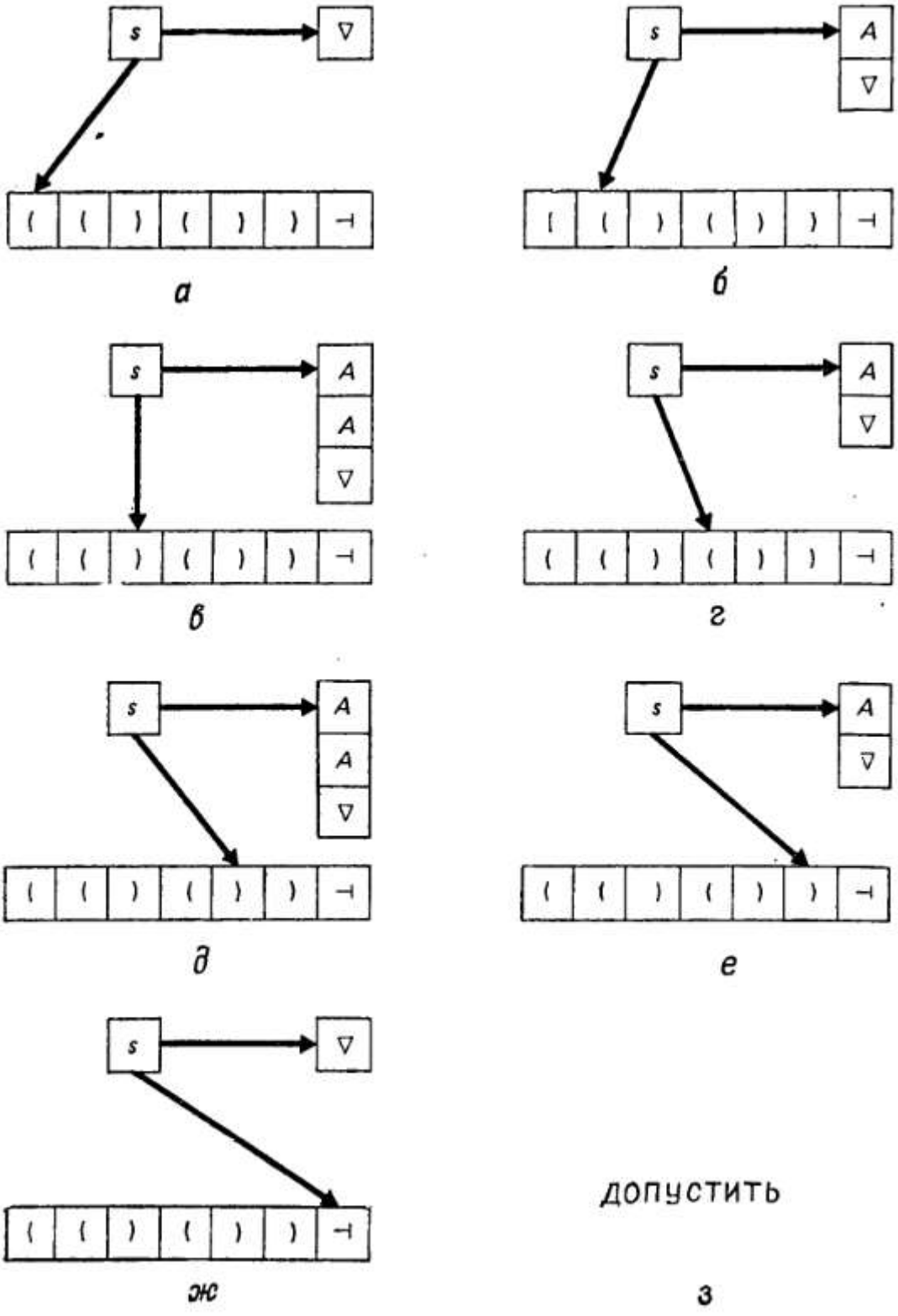

Рис. 5.3.

На рисунке показан каждый шаг процесса обработки, начиная с начальной конфигурации на рис. 5.3, *а* и кончая допускающей конфигурацией на рис. 5.3, *з*. Такое изображение последовательности конфигураций МП-автомата требует много места, поэтому представим ее в таком более компактном виде:

а: ▽	[*s*]	(()())⊣
б: ▽*A*	[*s*]	()())⊣
в: ▽*AA*	[*s*]	)())⊣
г: ▽*A*	[*s*]	())⊣
д: ▽*AA*	[*s*]	))⊣
е: ▽*A*	[*s*]	)⊣
ж: ▽	[*s*]	⊣
з: ДОПУСТИТЬ		

В этом линейном представлении конфигураций МП-автомата магазин изображен слева, состояние — в середине, а необработанная часть входной цепочки — справа. Эта часть входной цепочки включает текущий входной символ и символы, которые следуют после

( ) ⊣

A	ВТОЛКНУТЬ(*A*) СДВИГ	ВЫТОЛКНУТЬ СДВИГ	ОТВЕРГНУТЬ
▽	ВТОЛКНУТЬ(*A*) СДВИГ	ОТВЕРГНУТЬ	ДОПУСТИТЬ

Рис. 5.4.

него. Чтобы восстановить всю входную цепочку, нужно вернуться назад, к исходной конфигурации. Информацию, поступающую в управляющее устройство, выделить очень легко, так как символ, расположенный на верху магазина, находится непосредственно слева от состояния, а текущий входной символ — справа от него. Многие из МП-автоматов, применяемых на практике, имеют лишь одно состояние (как в этом примере), и в подобных случаях мы, как правило, опускаем информацию, касающуюся состояний.

Управляющее устройство этого автомата с одним состоянием можно представить в виде управляющей таблицы, как на рис. 5.4, где показаны действия автомата для каждого сочетания входного символа и верхнего символа магазина. Столбцы таблицы обозначены входными символами, а на пересечении строк и столбцов

обозначены соответствующие им действия. Так как этот конкретный автомат имеет лишь одно состояние, информация о состоянии опущена.

Мы будем пользоваться таблицами такого вида (т. е. со столбцами для входных символов и строками для символов магазина) как стандартным представлением МП-автоматов с одним состоянием.

5.3. Пример распознавания множества МП-автоматом

Чтобы привести еще один пример распознавания МП-автоматом, использующим более чем одно состояние, рассмотрим задачу распознавания множества

$$\{0^n 1^n | n > 0\}.$$

В качестве первого шага построения МП-распознавателя опишем словами схему распознавания:

Начальный отрезок цепочки, состоящий из нулей, вталкивается в магазин. Затем каждый раз, когда встречается единица, один нуль выталкивается из магазина. Цепочка допускается тогда и только тогда, когда в момент завершения считывания цепочки магазин пуст. Если после первого вхождения единицы встречается нуль, цепочка сразу отвергается.

1:	∇	$[s_1]$	0 0 0 1 1 1 ⊣
2:	∇ Z	$[s_1]$	0 0 1 1 1 ⊣
3:	∇ Z Z	$[s_1]$	0 1 1 1 ⊣
4:	∇ Z Z Z	$[s_1]$	1 1 1 ⊣
5:	∇ Z Z	$[s_2]$	1 1 ⊣
6:	∇ Z	$[s_2]$	1 ⊣
7:	∇	$[s_2]$	⊣
8:	ДОПУСТИТЬ		

а

1:	∇	$[s_1]$	0 0 1 0 1 1 ⊣
2:	∇ Z	$[s_1]$	0 1 0 1 1 ⊣
3:	∇ Z Z	$[s_1]$	1 0 1 1 ⊣
4:	∇ Z	$[s_2]$	0 1 1 ⊣
5:	ОТВЕРГНУТЬ		

б

Рис. 5.5.

Состояние 1	0	1	⊣
Z	СОСТОЯНИЕ (s_1) ВТОЛКНУТЬ (Z) СДВИГ	СОСТОЯНИЕ (s_2) ВЫТОЛКНУТЬ СДВИГ	ОТВЕРГНУТЬ
∇	СОСТОЯНИЕ (s_1) ВТОЛКНУТЬ (Z) СДВИГ	ОТВЕРГНУТЬ	ОТВЕРГНУТЬ

Состояние 2	0	1	⊣
Z	ОТВЕРГНУТЬ	СОСТОЯНИЕ (s_2) ВЫТОЛКНУТЬ СДВИГ	ОТВЕРГНУТЬ
∇	ОТВЕРГНУТЬ	ОТВЕРГНУТЬ	ДОПУСТИТЬ

Начальное содержимое магазина: ∇

Рис. 5.6.

5.4. Расширенные операции над магазином

На самом деле имеется много разных способов определения класса моделей автоматов, переходы которых выбираются в зависимости от входа, состояния и верхних магазинных символов, а операции затрагивают только верхнюю часть магазина. Каждую из этих моделей принято называть автоматом с магазинной памятью. Таким образом, модель автомата, описанная в разд. 5.1,— это как раз пример модели МП-автомата.

В тех случаях, когда может возникнуть недоразумение, мы будем называть МП-автоматы такого типа *примитивными МП-автоматами*.

В этом разделе мы введем расширенную операцию над магазином, назовем ее ЗАМЕНИТЬ и проиллюстрируем ее использование. Другие расширенные операции будут вводиться в последующих главах по мере надобности.

Операция ЗАМЕНИТЬ состоит в выталкивании верхнего символа магазина и последующем выполнении нескольких вталкиваний. Последовательность символов, которые операция ЗАМЕНИТЬ должна помещать в магазин, указывается в качестве ее аргумента. Так, мы пишем

ЗАМЕНИТЬ (*ABC*)

если в магазин нужно поместить *ABC*. Это эквивалентно последовательности операций

ВЫТОЛКНУТЬ
ВТОЛКНУТЬ(*A*)
ВТОЛКНУТЬ(*B*)
ВТОЛКНУТЬ(*C*)

Таким образом, левый символ последовательности помещается в магазин первым и оказывается ниже остальных символов этой последовательности. Если операция ЗАМЕНИТЬ (*ABC*) применяется к магазину

$$\nabla\ X\ Y\ Z$$

то новый магазин выглядит так:

$$\nabla\ X\ Y\ A\ B\ C$$

Операция ЗАМЕНИТЬ широко и систематически используется в последующих главах. В данный момент мы рассматриваем ее просто как сокращение для последовательности примитивных операций над магазином, которую программист может включить как часть одной процедуры перехода.

Чтобы проиллюстрировать использование операции ЗАМЕНИТЬ, вернемся к задаче распознавания множества $\{0^n1^n|n>0\}$. В разд. 5.3 мы видели, что можно построить распознаватель, который работает в двух фазах — «вталкивания» и «выталкивания». Мы строили такой автомат, пользуясь для запоминания фазы управляющим состоянием. Теперь для этой же задачи построим другой МП-автомат.

Новый МП-автомат использует тот же метод счета, что и предыдущий автомат. *Z* вталкивается в магазин при каждом появлении

на входе символа 0 и выталкивается из него при каждом появлении на входе символа 1. Однако для различения фаз вталкивания и выталкивания используется иная стратегия. Во время фазы вталкивания в верхней ячейке магазина хранится новый магазинный символ X. Единственное его назначение — напоминать управляющему устройству, что автомат находится в фазе вталкивания. Когда впервые встречается единица, X выталкивается из магазина и автомат начинает сопоставлять символы Z и единицы. Наличие процедуры ЗАМЕНИТЬ позволяет нам реализовать этот алгоритм с помощью единственного состояния, как показано на рис. 5.7. Последовательность конфигураций при распознавании цепочки 0 0 0 1 1 1 показана на рис. 5.8. Эту последовательность конфигураций можно сравнить с рис. 5.5, *а*, где изображена обработка этой же цепочки предыдущим автоматом.

	0	1	⊣
X	ЗАМЕНИТЬ (ZX) СДВИГ	ВЫТОЛКНУТЬ ДЕРЖАТЬ	ОТВЕРГНУТЬ
Z	ОТВЕРГНУТЬ	ВЫТОЛКНУТЬ СДВИГ	ОТВЕРГНУТЬ
∇	ОТВЕРГНУТЬ	ОТВЕРГНУТЬ	ДОПУСТИТЬ

Начальное содержимое магазина : ∇X

Рис. 5.7.

1:	∇X	0 0 0 1 1 1 ⊣
2:	$\nabla Z X$	0 0 1 1 1 ⊣
3:	$\nabla Z Z X$	0 1 1 1 ⊣
4:	$\nabla Z Z Z X$	1 1 1 ⊣
5:	$\nabla Z Z Z$	1 1 1 ⊣
6:	$\nabla Z Z$	1 1 ⊣
7:	∇Z	1 ⊣
8:	∇	⊣
9:	ДОПУСТИТЬ	

Рис. 5.8.

Операция ЗАМЕНИТЬ используется, когда на верху магазина X, а на входе 0. За один шаг эта операция выталкивает из магазина ненужный верхний символ X, помещает на его место символ для запоминания вхождения 0, а затем помещает на верх магазина другой X, чтобы указать, что автомат по-прежнему в «фазе вталкивания».

Построить примитивные МП-распознаватели языков.
Подобрать тестовые данные.
Записать протоколы работы МП-распознавателей на тестовых данных.

1) $L=\{1^n20^n \mid n>0\}$
2) $L=\{21^n20^{2n} \mid n>0\}$
3) $L=\{1^n20^n \mid n>0\} \cup \{21^n20^{2n} \mid n>0\}$
4) Все цепочки из 0 и 1, в которых количество 1 равно количеству 0.
5) $L=\{1^n20^n1^m20^m \mid n>0 \text{ и } m\geq 0\}$
6) $L=\{0^n1^m0^n \mid n>0 \text{ и } m>0\}$
7) $L=\{a^nb^mc^{2(n+m)} \mid n\geq 0 \text{ и } m\geq 0\}$
8) $L=\{0^n1^m \mid n\leq m\}$
9) $L=\{0^n1^m \mid n\geq m\}$
10) $L=\{10^n10^n1 \mid n\geq 0\}$
11) $L=\{0^n1^m \mid n\neq m \text{ и } n>0 \text{ и } m>0\}$
12) $L=\{a0^n1^n \mid n>0\} \cup \{b0^n1^{2n} \mid n>0\}$
13) $L=\{1^n0^n1^m0^m \mid n>0 \text{ и } m\geq 0\}$
14) $L=\{1^n0^m1^m0^n \mid n>0 \text{ и } m\geq 0\}$