LR-анализ

А. А. Рубцов

3 декабря 2018 г.

1 Магазинный автомат для правого вывода

Пусть G — КС-грамматика. Опишем недетерминированный (расширенный) МП-автомат, допускающий по пустому стеку, который распознаёт язык $L(G)$. Мы считаем, что стек автомата растёт слева направо. Автомат совершает три типа действий:

S: *сдвиг* — перемещения символа со входной ленты в стек;

R: *свёртка* — если на верхушке стека накопилась правая часть правила $A \to \alpha$, то автомат извлекает из стека цепочку α и заменяет её на A;

Acc: *допуск* — опустошение стека, если он содержит только аксиому (и окончание на этом работы).

В случае если слово обработано, и в результате действий автомата в стеке осталась только аксиома, то после действия **Acc** слово будет принято автоматом, поскольку стек окажется пуст.

МП-автоматы данного вида мы будем называть SR-автоматами.

Утверждение 1. *Каждая конфигурация*[1] (α, u) *SR-автомата на успешном ходе соответсвует обратному шагу* αu *правого вывода соответствующего слова. При действие* **R** *переход между конфигурациями соответсвует обратному шагу вывода.*

[1]Мы не уделяем внимания техническому устройству SR-автоматов, а посему считаем, что конфигурация состоит из содержимого стека и необработанной части входа. Также мы не обращаем внимание на маркер дна.

Доказательство. Докажем утверждение индукцией по обратному ходу автомата на слове w. База. Перед тем как принять слово, автомат находился в конфигурации (S, ε) — эта конфигурация соответсвует первому шагу вывода w. Шаг индукции. Пусть автомат находится в конфигурации (α, u) тогда есть два варианта: либо $\alpha = \beta A v, v \neq \varepsilon$, либо $\alpha = \beta A$. В первую конфигурацию автомат попал только совершив действие **S**, которое не нарушает соответствия между конфигурацией и шагом вывода, а во втором случае автомат совершил действие **R**, а значит совершил свёртку по некоторому правилу $A \to \gamma$. Поскольку A самый правый нетерминал текущего шага вывода, то предыдущая конфигурация МП-автомата соответствовала шагу правого вывода. Поскольку начальная конфигурация автомата — конфигурация (ε, w), то соответствующий ходу автомата правый вывод является выводом слова w. □

Следствие 1. *SR-автомат для грамматики G распознаёт язык $L(G)$.*

Следствие 2. *Последовательность номеров правил, по которым SR-автомат производил свёртку на успешном ходе, является правым выводом поданного на вход слова.*

SR-автоматы являются недетерминированными, но для некоторых грамматик возможна их детерминизация. Процедура детерминизации, которую мы приведём далее, применима для так называемых LR(k)-грамматик, а автомат полученный на выходе называется LR(k)-анализатором. Анализаторы, вообще говоря, в результате работы получают вывод поданного на вход слова, но мы не будем уделять этому вопросу внимание, поскольку получить вывод во время работы SR-автомата, которым по-сути является LR-анализатор нетрудно (следствие 2).

2 Обход в глубину дерева вывода МП-автоматом

Ключевое наблюдением для нашей техники детерминизации состоит в том, что с помощью SR-автомата можно реализовать обход дерева вывода в глубину.

При каждом шаге обхода в глубину корневого дерева, открытые вершины образуют путь, начинающийся в корне. Этот путь будет поддерживаться в стеке автомата в виде последовательности открытых вершин. Для этого, помимо описания вершины, нам потребуется хранить её родителя и «сестёр» — вершин, имеющих с описываемой вершиной общего родителя. Эту информацию мы будем описывать следующим образом. Пусть вершина Z оказалась в дереве в результате применения правила $A \to X_1 X_2 \dots X_l Z Y_1 \dots Y_r$, где $X_i, Y_i, Z \in N \cup \Sigma$. Тогда

в качестве описания Z мы будем хранить набор

$$[A \to X_1 \dots X_l \cdot Z Y_1 \dots Y_r],$$

который называется LR(0)-ситуацией. Смысл названия мы объясним позже, а пока будем называют этот набор просто *ситуацией*. В случае описания корня дерева, мы будем использовать ситуацию $[S' \to \cdot S]$, здесь S' — вспомогательный нетерминал, нужный нам для технических целей.

Пример 1. *Грамматика G задана правилами $S \to Ab \mid \varepsilon;\ A \to aSb$. Для дерева вывода слова abb обход в глубину будет поддерживаться следующим образом.*

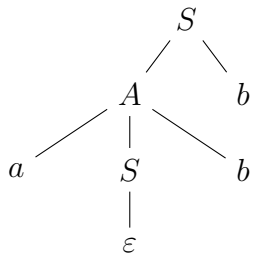

Рис. 1: Дерево вывода слова abb

Протокол работы автомата, реализующего обход в глубину приведён в таблице 2. Мы приведём определение этого автомата ниже, а пока предлагаем читателю познакомиться с тем как мы будем хранить описание обхода в глубину в стеке.

Под ε-действием мы понимаем добавление в стек ситуации, описывающей вершину, которая получается при спуске из последней открытой вершине в поиске в глубину. Строки таблицы, в которых не описано действие, являются промежуточным шагом при переходе от одной конфигурации автомата к другой и мы вставили их для наглядности. Определим теперь формально DFS-автомат.

DFS-автомат совершает те же действия, что и SR-автомат, но помимо этого хранит в стеке ситуации и совершает ε-действия, которые мы определим чуть позже. На вершине стека DFS-автомата всегда находится LR(0)-ситуация, которая и определяет его действие. Помимо LR(0)-ситуаций, описывающих открытые вершины, мы будем использовать LR(0)-ситуации вида $[A \to X_1 \dots X_n \cdot]$, которые означают, что все дочерние вершины A были закрыты поиском в глубину. Опишем действие автомата в зависимости от вида ситуации. Мы предполагаем,

Шаг поиска в глубину	***Стек***	***действие***	***Шаг обратного правого вывода***
S	$[S' \to \cdot S]$	ε	abb
$S \to A$	$[S' \to \cdot S][S \to \cdot Ab]$	ε	abb
$S \to A \to a$	$[S' \to \cdot S][S \to \cdot Ab][A \to \cdot aSb]$	**S**	abb
$S \to A \to a$	$[S' \to \cdot S][S \to \cdot Ab][A \to \cdot aSb]a$		abb
$S \to A \to S$	$[S' \to \cdot S][S \to \cdot Ab]a[A \to a \cdot Sb]$	ε	abb
$S \to A \to S \to \varepsilon$	$[S' \to \cdot S][S \to \cdot Ab]a[A \to a \cdot Sb][S \to \cdot]$	**R**	abb
$S \to A \to S$	$[S' \to \cdot S][S \to \cdot Ab]a[A \to a \cdot Sb]S$		abb
$S \to A \to b$	$[S' \to \cdot S][S \to \cdot Ab]aS[A \to aS \cdot b]$	**S**	$aSbb$
$S \to A \to b$	$[S' \to \cdot S][S \to \cdot Ab]aSb[A \to aSb\cdot]$	**R**	$aSbb$
$S \to A$	$[S' \to \cdot S][S \to \cdot Ab]A$		Ab
$S \to b$	$[S' \to \cdot S]A[S \to A \cdot b]$	**S**	Ab
$S \to b$	$[S' \to \cdot S]Ab[S \to Ab\cdot]$	**R**	Ab
S	$[S' \to \cdot S]S$		S
	$S[S' \to S\cdot]$	**Acc**	

Рис. 2: Протокол работы DFS-автомата на слове abb

что автомат недетерминированно угадывает следующий шаг обхода в глубину и нужную информацию о дереве вывода, исходя из чего принимает действия, соответсвующие обходу в глубину.

(i) $[A \to \alpha \cdot B\beta]$: ε-действие. В случае, если B имеет дочерние вершины $X_1 \dots X_n$, автомат добавляет в стек ситуацию $[B \to \cdot X_1 \dots X_n]$.

(ii) $[A \to \alpha \cdot a\beta]$: **S**. Ситуация извлекается из стека, а вместо неё в стек добавляется ситуация $a[A \to \alpha a \cdot \beta]$.

(iii) $[A \to \alpha \cdot], A \neq S'$: **R**. Такая ситуация возможна, только если на вершине стека находится

$$[B \to \beta \cdot A\gamma][A \to \cdot\alpha]\alpha[A \to \alpha \cdot];$$

это слово заменяется на $A[B \to \beta A \cdot \gamma]$.

(iv) $[S' \to S \cdot]$: **Acc**.

В начале работы DFS-автомата в стеке лежит ситуация $[S' \to \cdot S]$.

Лемма 1. *Содержимое стека DFS-автомата на каждом шаге соответсвует шагу обхода в глубину дерева вывода слева направо.*

Доказательство. Докажем индукцией по шагу вычисления DFS-автомата. База: $[S' \to \cdot S]$ соответсвует тому, что открыта вершина S. Шаг индукции. По предложению индукции, случай (i) означает, что последняя открытая вершина нетерминальная, случай (ii) означает, что открыт лист, а случай (iii) означает, что все дочерние вершины вершины A были закрыты. Первому случаю соответсвует спуск при обходе в глубину, а остальным подъём — действия автомата меняют содержимое стека сохраняя биекцию между шагом хода в глубину и его кодировкой в стеке, согласно выбранному описанию. Заметим, что действие (iv) является частным случаем (iii): Если в стеке лежит ситуация $[S' \to S \cdot]$, то перед ней находится аксиома S. □

Действия DFS-автомата определяются видом ситуации на вершине стека. Заметим, что в результате сдвига или свёртки происходит переход от ситуации $[A \to \alpha \cdot X\beta]$ к $[A \to \alpha X \cdot \beta]$ с удалением первой, а в случае ε-действия после ситуации $[A \to \alpha \cdot B\beta]$ может идти любая ситуация вида $[B \to \cdot\gamma]$ (без удаления первой). Эти переходы удобно задать в виде графа недетерминированного автомата, который мы будем называть недетерминированным LR(0)-автоматом или недетерминированным автоматом Кнута. Формально, множество состояний

автомата образуют ситуации, достижимые из ситуации $[S' \to \cdot S]$, которая является начальным состоянием автомата; автомат имеет два вида переходов:

$$[A \to \alpha \cdot X\beta] \xrightarrow{X} [A \to \alpha X \cdot \beta] \quad \text{и} \quad [A \to \alpha \cdot B\beta] \xrightarrow{\varepsilon} [B \to \cdot\gamma].$$

Множество принимающих состояний не определено — для наших нужд оно не потребуется.

3 LR-анализаторы

Мы переходим к определению LR(k)-анализаторов. Основная идея состоит в следующем. Если детерминизировать недетерминированный конечный автомат (автомат Кнута), то для хорошей грамматики получится детерминизировать и DFS-автомат. Изобретатель LR-анализа, Дональд Кнут, исходил из схожей идеи, но использовал немного другой подход.

3.1 От DFS к DFS(k)

Точнее, нам потребуется немного изменить DFS-автомат под технические нужды, которые во многом определяются расшифровкой названия «LR(k)». L означает, что анализатор (детерминированный МП-автомат) читает слово слева направо, R — что восстанавливает правый вывод, а k — это количество символов, нужное анализатору для предпросмотра, чтобы принять решение какое действие выбрать.

В случае $k \geqslant 1$ анализатор получает также информацию о наступлении конца слова — для этого используется маркер конца слова \$, такми образом, на вход анализатора подаётся слово w\$.

Чтобы определить LR(k)-анализатор, мы определим сначала DFS(k)-автомат, который отличается от DFS-автомата техническими деталями, которые не нарушают доказанных свойств, но позволяют построить по DFS(k)-автомату LR(k)-анализатор.

Аналогично DFS-автомату, работу DFS(k)-автомата будут определять LR(k)-ситуации. LR(0)-ситуации мы уже определили выше; LR(k)-ситуацией для $k \geqslant 1$ будем называть набор $[A \to \alpha \cdot X\beta, u]$, где α и β произвольные цепочки из нетерминалов и терминалов, X — либо нетерминал, либо терминал, либо пустое слово (если только $\alpha = \beta = \varepsilon$), а u — слово длины от 1 до k.

Слово u называют аванцепочкой, смысл которой — префикс необработанной части входа после свёртки по правилу $A \to \alpha X \beta$ после обхода всего поддерева,

растущего из A. Смысл остальных частей LR(k)-ситуации такой же, как и у LR(0)-ситуации — легко видеть, что определение LR(0)-ситуации получается из определения в общем случае при $k = 0$.

Итак, DFS(k)-автомат действует точно также как и DFS-автомат, только перед тем как совершить свёртку, он проверяет, что необработанная часть входа совпадает с аванцепочкой ситуации, а перед тем как выполнить сдвиг из ситуации $[A \to \alpha \cdot a\beta, u]$, автомат проверяет, что некоторый префикс необработанной части входа принадлежит множеству $\mathrm{FIRST}_k(a\beta u)$. Конечно, для недетерминированного автомата это действия абсолютно лишние, но они будут важны для его детерминизации.

Также для детерминизации будет важно следующее техническое условие на содержимое стека. Мы определили DFS автомат так, чтобы ситуации в его стеке соответствовали пути из всех открытых вершин в поиске в глубину на данном шаге. Для этого, мы удаляли из стека ситуацию $[A \to \alpha \cdot a\beta]$ при сдвиге a, — теперь мы не будем этого делать. То есть, при сдвиге a на верхушке стека будет

$$[A \to \alpha \cdot a\beta, u]a[A \to \alpha a \cdot \beta, u].$$

Это не испортит работы автомата — при свёртке по правилу $A \to X_1 \dots X_n$ будем из стека извлекать не $X_1 \dots X_n[A \to X_1 \dots X_n \cdot]$, а

$$[A \to \cdot X_1 \dots X_n, u]X_1[A \to X_1 \cdot X_2 \dots X_n, u] \dots X_n[A \to X_1, \dots X_n \cdot, u].$$

Теперь, содержимое стека DFS(k)-автомата имеет вид $S_0Y_1S_1 \dots S_{n-1}Y_nS_n$, где S_i — непустая последовательность из LR(k)-ситуаций, причём в S_n входит ровно одна ситуация, которая и определяет действие, как описано выше.

Аналогично недетерминированному LR(0)-автомату, по которому однозначно строится управляющая таблица DFS-автомата, определим недетерминированный LR(k)-автомат, который также будем называть автоматом Кнута. Начальное состояние автомата — ситуация $[S' \to \cdot S, \$]$; автомат имеет два вида переходов:

$$[A \to \alpha \cdot X\beta, u] \xrightarrow{X} [A \to \alpha X \cdot \beta, u] \quad \text{и} \quad [A \to \alpha \cdot B\beta, u] \xrightarrow{\varepsilon} [B \to \cdot\gamma, v],$$

для всех правил $B \to \gamma$ и всех слов $v \in \mathrm{FIRST}_k(\beta u)$.

Из сказанного выше следует, что LR(k)-автомат однозначно определяет управляющую таблицу DFS(k)-автомату.

3.2 Определение LR(k)-анализатора

Мы определяем LR(k)-анализатор для грамматики G по автомату, полученному из недетерминированного LR(k)-автомата для G путём его детерминизации. Получившийся автомат мы будем называть детерминированным LR(k)-автоматом или детерминированным автоматом Кнута.

Аналогично DFS(k)-автомату, LR(k)-анализатор будет SR-автоматом, стек которого на каждом шаге работы имеет вид

$$I_0 Y_1 I_1 \dots I_{n-1} Y_n I_n,$$

здесь I_j — всегда одно состояние детерминированного автомата Кнута, в отличие от последовательности S_j состояний недетерминированного автомата Кнута. LR(k)-автомат принимает решение о применении действий сдвиг или свёртка исходя из состояния I_n на верхушке стека и k символов необработанной части входа в соответствующей конфигурации SR-автомата. Мы считаем, что анализатор предпросматривает эти символы, не забирая их со входной ленты. Ключевое отличие LR(k)-анализатора от приведённых выше автоматов в том, что он детерминированный SR-автомат.

Опишем как по детерминированному автомату Кнута задать управляющую таблицу детерминированного SR-автомата. Оговоримся сразу, что это не всегда можно сделать — если описанная ниже процедура корректно определяет управляющую таблицу, то грамматика G называется LR(k)-грамматикой.

Обозначим функцию, определяющую действие автомата через $\mathsf{Action}(I, v)$, где I — состояние автомата Кнута, а v — непустое слово над алфавитом $\Sigma \cup \{\$\}$ длины, не более k.

- $\mathsf{Action}(I, v) = \mathbf{S}$, если I содержит ситуацию $[A \to \alpha \cdot a\beta, u]$ и $v \in \mathrm{FIRST}_k(a\beta u)$;
- $\mathsf{Action}(I, v) = \mathbf{R}(A \to \alpha)$, если I содержит ситуацию $[A \to \alpha \cdot, v]$;
- $\mathsf{Action}(I, \$) = \mathbf{Acc}$, если I содержит ситуацию $[S' \to S \cdot, \$]$.

Теперь, при действии $\mathbf{R}$ мы указывая правило, по которому происходит свёртка, потому что в одном состоянии I могут оказаться ситуациивида $[A \to \alpha \cdot, u]$ и $[B \to \beta\alpha \cdot, v]$ и анализатор должен совершить свёртку только по одному из правил, что невозможно при $u = v$. Дальше мы будем считать, что правило указано и при свёртки DFS(k)-автомата.

В результате каждого действия, в стеке оказывается некоторый символ X (терминал a в случае сдвига и нетерминал A в случае свёртки) после чего на

верхушку стека записывается состояние J, такое что в детерминированном автомате Кнута есть переход $I \xrightarrow{X} J$.

Заметим, что функция Action определена так, что если состояние I, которое является множеством LR(k)-ситуаций, содержит ситуацию s с аванцепочкой u, то действие $\mathsf{Action}(I, u)$ совпадает с действием DFS(k)-автомата с ситуацией s на верхушке стека, если s определяет не ε-действие.

Определение 1. Грамматика G называется LR(k)-грамматикой, если функция Action (определяемая по детерминированному LR(k)-автомату для грамматики G) определена корректно.

Теорема 1. *Пусть G — LR(k)-грамматика. Тогда LR(k)-анализатор распознаёт язык $L(G)$ и по протоколу анализатора однозначно восстанавливается правый разбор успешно обработанного слова.*

Доказательство. Поскольку успешный ход LR(k)-анализатора совпадает с успешным ходом SR-автомата, по следствию 1, он может принять только слова из языка $L(G)$; при этом, согласно следствию 2, по протоколу работы LR(k) анализатора однозначно восстанавливается правый вывод принятого слова.

Докажем теперь, что анализатор распознаёт все слова языка $L(G)$. Рассмотрим DFS(k)-автомат по которому был построен анализатор. По лемме 1, каждому обходу в глубину соответсвует ход DFS(k)-автомата, а значит он распознаёт язык $L(G)$, и кроме того на каждом ходе содержимое стека описывает шаг обхода в глубину в указанной кодировке. Докажем, что анализатор также имеет успешный ход на каждом слове из языка $L(G)$.

Зафиксируем слово $w \in L(G)$ и успешный ход DFS(k)-автомата на нём. Будем строить успешный ход LR(k)-анализатора по ходу автомата. Формально, мы докажем индукцией по длине хода, что содержимые стека автомата и анализатора на каждом шаге имеют соответсвенно вид

$$S_0 Y_1 S_1 \dots S_{n-1} Y_n S_n \quad \text{и} \quad I_0 Y_1 I_1 \dots I_{n-1} Y_n I_n,$$

причём каждое состояние s последовательности S_i принадлежит множеству I_i.

Заметим, что если S_i — последовательность ситуаций $s_1, s_2, \dots, s_m$, то в недетерминированном автомате Кнута есть переходы $s_1 \xrightarrow{\varepsilon} s_2 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} s_m$, а значит каждая ситуация s_j принадлежит множеству I_i, если $s_1 \in I_i$ — при детерминизации, если $s \in I$, то и все состояния, достижимые из s по ε-переходам оказываются в I.

<u>База:</u> начальные конфигурации S_0 и I_0, где $S_0 = [S' \to \cdot S, \$]$, удовлетворяют условию, поскольку $S_0 \subseteq I_0$ по построению детерминированного автомата

Кнута. Шаг индукции: если автомат делает ε-действие, то последовательность S_n пополняется ситуацией s, достижимой по ε-переходу в недетерминированном автомате Кнута из последней ситуации в S_n, — значит $s \in I_n$.

В случае сдвига или свёртки, в стек попадает символ X; в случае свёртки по правилу $X \to Y_1 \dots Y_l$, перед добавлением X, с верхушек стека автомата и анализатора удаляются соответсвенно цепочки

$$Y_1 S_{n-l+1} \dots Y_{l-1} S_{n-1} Y_l S_n \quad \text{и} \quad Y_1 I_{n-l+1} \dots Y_{l-1} I_{n-1} Y_l I_n.$$

Итак, символ X добавляется в стек автомата, когда на верхушке стека лежит последовательность S_j, где $j = n$ в случае сдвига и $j = n - l$ в случае свёртки. Последняя ситуация последовательности S_j имеет вид $s = [A \to \alpha \cdot X\beta, u]$ и $s \in I_j$ по предложению индукции. После добавления X, в стек автомата добавляется ситуация $s' = [A \to \alpha \cdot X\beta, u]$. В недетерминированном автомате Кнута есть переход $s \xrightarrow{X} s'$, а значит $s' \in I_{j+1}$, где $I_j \xrightarrow{X} I_{j+1}$ (в детерминированном автомате Кнута). Шаг индукции доказан.

Получаем, что если DFS(k)-автомат совершил действие **Acc**, то его же совершил и LR(k)-анализатор. □

Заметим, что если G — LR(k)-грамматика, то G и LR($k+1$)-грамматика. Но как проверить, что G является или не является LR(k)-грамматикой, глядя на управляющую таблицу LR($k+1$)-анализатора?

Чтобы ответить на этот вопрос, обратимся к процедуре детерминизации автомата Кнута. Рассмотрим детерминированные LR(k)- и LR($k+1$)-автоматы для грамматики G (которая не обязательно LR(k) или LR($k+1$)). Будем обозначать через I_ξ состояние детерминированного автомата Кнута, достижимое из I_0 по ξ: $I_0 \xrightarrow{\xi} I_\xi$.

Утверждение 2. *Пусть I_ξ — состояние* LR($k+1$)*-автомата. Тогда для каждой ситуации $s \in I_\xi$ с аванцепочкой u в в состоянии J_ξ* LR(k)*-автомата есть ситуация s', отличающаяся от s лишь быть может аванцепочкой v и v — префикс u. С другой стороны, для каждой ситуации $s' \in J_\xi$ существует ситуация $s \in I_\xi$ с той же аванцепочкой v или с аванцепочкой u с префиксом v.*

Доказательство. Докажем утверждение индукцией по длине ξ. База: $I_\varepsilon = I_0$. В состояние I_0 и J_0 входят ситуации, достижимые из начальной ($[S' \to \cdot S, \$]$) в графе соответствующего недетерминированного автомата Кнута по ε-переходам. Начальные ситуации в I_0 и J_0 либо совпадают, при $k \geqslant 0$, либо аванцепочка

пуста в случае $k = 0$ — для начальной ситуации I_0 утверждение верно. Пусть теперь $s_1 \xrightarrow{\varepsilon} s$ и для $s_1 \in I_0$ существует $s'_1 \in J_0$. Докажем, что тогда в J_0 найдётся подходящая ситуация s'_1.

Действительно, ε-переход имеет вид $s_1 = [A \to \alpha \cdot B\beta, u_1] \xrightarrow{\varepsilon} [B \to \cdot\gamma, u], u \in \mathrm{FIRST}_{k+1}(\beta u_1)$, но тогда в недетерминированном $\mathrm{LR}(k)$-автомате есть ε-переход $s'_1 = [A \to \alpha \cdot B\beta, v_1] \xrightarrow{\varepsilon} [B \to \cdot\gamma, v], v \in \mathrm{FIRST}_k(\beta v_1)$, а значит существует слово v, которое является префиксом u. С другой стороны понятно, что в процессе ε-замыкания в J мы не можем получить ситуацию с аванцепочкой v, которая не будет префиксом аванцепочки соответствующей ситуации в I.

Шаг индукции. Пусть для всех ξ длины не больше, чем n утверждение верно. Тогда докажем утверждение для состояния $I_{\xi X}$: $I_\xi \xrightarrow{X} I_{\xi X}$. По алгоритму детерминизации в $I_{\xi X}$ попадают состояния вида $s = [A \to \alpha X \cdot \beta, u]$, где $s_1 = [A \to \alpha \cdot X\beta, u] \in I_\xi$, и достижимые из них по ε-переходам. Но по предложению индукции, для $s \in I_\xi$ существует $s' \in J_\xi$, а значит существует $s'_1 \in J_{\xi X}$: $s' \xrightarrow{X} s'_1$. При доказательстве базы, мы доказали, что в результате ε-замыкания для каждого s найдётся s'. Повторяя те же аргументы, только теперь смотря на $s' = [A \to \alpha X \cdot \beta, v]$, где $s'_1 = [A \to \alpha \cdot X\beta, v] \in J_\xi$, получаем справедливость второй части утверждения. □

Лемма 2. *Пусть G — $\mathrm{LR}(k{+}1)$-грамматика. Грамматика G не является $\mathrm{LR}(k)$-грамматикой, тогда и только тогда, когда в некотором множестве ситуаций I детерминированного $\mathrm{LR}(k+1)$-автомата есть пары ситуаций вида*

(i) $[A \to \alpha\cdot, ua]$ *и* $[B \to \xi\cdot, ub]$, $|u| = k$;

(ii) $[A \to \alpha \cdot a\beta, ub]$ *и* $[B \to \xi\cdot, vc]$, *где* $v \in \mathrm{FIRST}_k(a\beta u)$

Доказательство. Докажем сначала, что если такие пары ситуаций есть, то G — не $\mathrm{LR}(k)$-грамматика. В первом случае, по утверждению 2 детерминированный $\mathrm{LR}(k)$-автомат содержит в некотором состоянии J ситуации $[A \to \alpha\cdot, u]$ и $[B \to \xi\cdot, u]$, а значит функция Action не определена корректно на паре (I, u), поскольку анализатор должен совершить свёртку по разным правилам. Во втором случае, функция Action не будет корректно определена на паре (I, v): J будет содержать ситуацию $[A \to \alpha\cdot a\beta, u]$, согласно которой нужно сделать сдвиг и ситуацию $[B \to \xi\cdot, v]$, согласно которой нужно будет сделать свёртку.

Докажем теперь, что если пар ситуаций указанного вида нет, то G — $\mathrm{LR}(k)$-грамматика. Допустим это не так, значит в некотором состоянии J детерминированного автомата есть две ситуации, которые приводят к конфликту при попытке определить функцию Action. Конфликты могут быть только двух видов: свёртки по разным правилам или сдвиг и свёртка. В первом случае, в J

есть ситуации вида $[A \to \alpha\cdot, u]$ и $[B \to \xi\cdot, u]$, $|u| = k$. Но тогда в соответствующем I ($J = J_\xi, I = I_\xi$), по утверждению 2 должны быть ситуации вида (i). Аналогично получаем, что в I должны быть ситуации вида (ii) при конфликте сдвиг-свёртка. $\square$