Теория и реализация языков программирования

Курочкин В.М., Столяров Л.Н.,
Сушков Б.Г., Флёров Ю.А.

Предисловие

На протяжении ряда лет студентам факультета управления и прикладной математики МФТИ читался курс «Языки программирования». Содержание этого курса постепенно установилось и в настоящее время он состоит из 4 разделов: Методы порождения языков программирования, грамматики, автоматы с магазинной памятью - глава 1; методы синтаксического анализа языка, использующие грамматики и автоматы - глава 2; методы перевода из одного языка на другой, в частности, перевод с помощью магазинного преобразователя и перевод, формально определяемый синтаксисом и семантикой («синтаксически управляемая трансляция») - глава 3; наконец, демонстрация на примере языка АЛГОЛА-60 методов реализации основных понятий и конструкций языка программирования на ЭВМ.

Как видно из этого перечисления, курс содержит не описание каких-либо конкретных языков программирования, а их теорию и по характеру изложения являются математическим, Исключение представляет лишь 4-ый раздел, носящий несколько «эвристический» характер и содержащий изложение конкретного способа реализации конкретного же языка (АЛГОЛ-60). По целому ряду соображений мы считали целесообразным включить этот материал в данный курс.

В МФТИ «языки программирования» читаются на 2-м курсе. Предполагается, что студенты знают к этому времени основные элементы языка АЛГОЛ-60, знакомы с ЭВМ (например, писали программы на АЛГОЛе-60 и языке ассемблера и пускали их на машине).

В оформлении книги большую помощь оказали мне преподаватели МФТИ *Борис Григорьевич Сушков* (глава I), *Юрий Арсеньевич Флёров* (глава II) и *Лев Николаевич Столяров* (глава III), просмотревшие конспекты лекций, написавших связный текст соответствующих глав, добавивших ряд примеров, поясняющих изложение, и задач.

Без их содействия издание этих лекций существенно бы задержалось. Приношу им большую благодарность.

В.М.Курочкин

Москва, 1973 г.

Оглавление

Глава 1.

Определение формальных языков

1.1. Порождающие грамматики

Предметом нашего рассмотрения будут *формальные языки*, т.е. множества цепочек (называемых также словами, правильными предложениями), состоящих из элементов некоторого конечного множества (множества основных или терминальных символов, терминального алфавита). Формальный язык обладает синтаксисом и семантикой.

Синтаксис формального языка - это совокупность признаков, позволяющая выделить правильные предложения из множества всех цепочек терминальных символов.

Семантика языка - это смысл, вкладываемый в предложение языка.

Применительно к машинным языкам (алгоритмическим языкам, языкам программирования) под синтаксисом подразумевают описание совокупности правильно написанных (на данном языке программирования) программ, а под семантикой - описание действий вычислительной машины.

Одним из средств задания синтаксиса языка являются формальные (порождающие) грамматики.

Определение 1.1.1. *Порождающей грамматикой* называется упорядоченная четвёрка $G = (V, T, P, S)$, где
V - конечное множество символов (алфавит грамматики),
$T \subseteq V$ - конечное множество терминальных символов (терминальный алфавит),
P - конечное множество упорядоченных пар вида $\varphi \to \psi$ (множество правил вывода грамматики или подстановок).

Здесь φ и ψ - произвольные цепочки в алфавите V, т.е. $\varphi, \psi \in V^*$. V^* будет обозначать множество всех цепочек в алфавите V.

S - начальный символ (аксиома) из V.

Множество $V_n = V - T$ называется нетерминальным (вспомогательным) алфавитом, а его элементы - вспомогательными (нетерминальными) символами.

Несколько соглашений относительно обозначений.

В дальнейшем будем использовать начальные строчные буквы латинского алфавита $a, b, c, ...$ для обозначения отдельных терминальных символов.

Заглавными начальными буквами латинского алфавита $A, B, C...$ будем обозначать символы, принадлежащие вспомогательному алфавиту.

На место начальных букв греческого алфавита $\alpha, \beta, \gamma, ...$ в формулах могут быть подставлены как символы нетерминального (вспомогательного), так и основного алфавита.

Для обозначения цепочек из терминальных символов используются конечные малые буквы латинского алфавита $\dots, u, v, w, x, y, z$, для цепочек из нетерминальных символов - соответствующие заглавные латинские буквы $\dots, U, V, W, X, Y, Z$.

Цепочки, в которые могут входить любые символы алфавита V, будем обозначать последними буквам греческого алфавита $\dots, \sigma, \tau, \varphi, \chi, \upsilon, \psi, \omega$.

Буква S обычно используется для обозначения аксиомы грамматики.

Символ $\varnothing$ обозначает пустую цепочку. Иногда вместо «порождающая грамматика» будем писать «грамматика».

Пример 1.1.2. $G = (\{a, b, c, S, A\}, \{a, b, c\}, P, S)$, где

$P = \{S \to ab, S \to bSAaA, S \to \varnothing, A \to bScb, SA \to Aab\}$

$G = (\{a, b, c, A, B, S\}, \{a, b, c\}, P, S)$,

$P = \{S \to AB, AB \to AaB, AaB \to AbcB, Ab \to ab, cb \to ca\}$;

G - порождающая грамматика.

Рассмотрим теперь, как применяются правила.

Определение 1.1.3. Пусть $G = (V, T, P, S)$ - порождающая грамматика и $\chi, \omega \in V^*$.

Будем писать $\chi \Rightarrow \omega$, если существуют σ, τ, φ и $\psi \in V^*$, такие, что $\chi = \sigma\varphi\tau$, $\omega = \sigma\psi\tau$ и $\varphi \to \psi \in P$.

Далее пишем $\chi \overset{*}{\Rightarrow} \omega$, если либо $\chi = \omega$, либо существуют $\omega_0, \omega_1, \ldots, \omega_r$, такие, что $\omega_0 = \chi$, $\omega_r = \omega$ и $\omega_i \Rightarrow \omega_{i+1}$ для всех i.

Последовательность $\omega_1, \ldots, \omega_r$ называется *выводом* ω_r из ω_0 (длины r) и записывается в виде

$\omega_0 \Rightarrow \omega_1 \Rightarrow \ldots \Rightarrow \omega_r$.

Определение 1.1.4. Пусть $G = (V, T, P, S)$ - порождающая грамматика. Множество цепочек

$L(G) = \{x \in T^* | S \overset{*}{\Rightarrow} x\}$

называется *языком*, порождаемым грамматикой G.

Иногда на вид правил $\varphi \to \psi$ порождающей грамматики накладываются ограничения

$$\varphi \in (V - T)^* - \{\varnothing\} \tag{1}$$

Покажем, что такое ограничение не меняет класса языков, порождаемых грамматиками.

Теорема 1.1.5. Для любой грамматики G_1 существует грамматика G_2, удовлетворяющая ограничению (1), такая, что $L(G_1) = L(G_2)$.

Доказательство. Пусть $G_1 = (V, T, P, S)$. Построим эквивалентную грамматику G_2. Поставим в соответствие каждому символу $\alpha \in V$ символ $\alpha' \in V'$, символу пустой цепочки $\varnothing$ поставим в соответствие символ $A (A \notin V_N \cup T)$.

Обозначим $V_2 = V' \cup T \cup \{A\}$.

Тогда $G_2 = (V_2, T, P_2, S')$,

где $P_2 = \{\varphi' \to \psi'$, если $\varphi \to \psi \in P$, $\varphi \neq \varnothing$,

$\varphi' \cup \psi'$ получаются из φ и ψ
в силу соответствия $V' \leftrightarrow V\}$
$\cup \{A \to \psi'$, если $\varphi \to \psi \in P$;
ψ' получается из ψ как выше $\}$
$\cup \{\alpha' \to \alpha A | \alpha' \in V'\}$
$\cup \{\alpha' \to A\alpha' | \alpha' \in V'\}$
$\cup \{a' \to a | a \in T\}$.

Ясно, что каждому выводу $S \overset{*}{\Rightarrow} x$ в G можно поставить в соответствие эквивалентный (т.е. заканчивающийся той же терминальной цепочкой) вывод в G_2. Для этого в первом выводе ставим всюду штрихи, в нужных местах помещаем символ A, затем все штрихи снимаем, применяя соответствующие правила из P_2. Таким образом, $L(G_1) \subseteq L(G_2)$. Пусть существует вывод $S' \overset{*}{\Rightarrow} x$ в G_2. Сотрём все штрихи. Исключим выводы вида $a \to a$ (получившиеся из $a' \to a$) и $\alpha \to \alpha A, \alpha \to A\alpha$ (получившиеся в результате применения правил $\alpha' \to \alpha' A$ и $\alpha' \to A\alpha'$). Получаем эквивалентный вывод в G_1, т.е. $L(G_1) = L(G_2)$. Ч.т.д.

1.2. Языки и грамматики непосредственных составляющих

Следующее ограничение приводит к возникновению семейства формальных языков, называемых языками непосредственных составляющих.

Определение 1.2.1. Грамматика $G = (V, T, P, S)$ называется грамматикой непосредственных составляющих (НС-грамматикой), если каждое правило имеет вид

$$\varphi A \psi \to \varphi \omega \psi$$

где $\varphi, \psi \in V^*$, $A \in V - T$, $\omega \in V^* - \{\varnothing\}$.

Иногда условие $\varphi, \psi \in V^*$ заменяется условием
$\varphi, \psi \in (V - T)^*$.

Легко показать, что это ограничение не меняет класса языков, порождаемых НС-грамматиками.

Определение 1.2.2. Длинной некоторой цепочки χ будем называть число символов в ней и обозначать эту величину $|\chi|$. Грамматика G называется *неукорачивающей*, если в каждом её правиле длина левой части не больше длины правой.

Покажем, что для любой неукорачивающей грамматики $G = (V, T, P, S)$ существует эквивалентная НС-грамматика.

Эта грамматика строится следующим способом.

Для каждого правила $\varphi \to \psi, \varphi = \alpha_1, \ldots, \alpha_k$, слово ψ разобъём на k штук непустых подслов $\psi = \psi_1\psi_2 \ldots \psi_k$.

Введём k штук новых нетерминальных символов $C_1, \ldots, C_k$, своих для каждого правила.

Заменим правило $\varphi \to \psi$ набором правил

$\alpha_1\alpha_2 \ldots \alpha_k \to C_1\alpha_2 \ldots \alpha_k$

$C_1\alpha_2 \ldots \alpha_k \to C_1C_2\alpha_3 \ldots \alpha_k$

...

$C_1 \ldots C_{k-1}\alpha_k \to C_1 \ldots C_{k-1}C_k$

$C_1C_2 \ldots C_k \to \psi_1C_2 \ldots C_k$

$\psi_1C_2 \ldots C_k \to \psi_1\psi_2C_3 \ldots C_k$

$\psi_1\psi_2 \ldots \psi_{k-1}C_k \to \psi_1\psi_2 \ldots \psi_{k-1}\psi_k$.

Эквивалентность так построенной НС-грамматики и исходной неукорачивающей грамматики G очевидна.

Таким образом класс неукорачивающих грамматик и класс грамматик непосредственных составляющих порождают одно и то же множество языков (НС-языков).

1.3. Контекстно-свободные языки и грамматики

Класс языков, к изучению которого мы сейчас приступаем, весьма важен, поскольку эти языки являются хорошими моделями для большинства употребляемых в настоящее время языков программирования.

Определение 1.3.1. Грамматика $G = (V, T, P, S)$ называется *контекстно-свободной* (КС-грамматикой), если каждое её правило имеет вид

$A \to \varphi$, где $A \in V - T$, $\varphi \in V^*$.

Термин «контекстно-свободная» в названии грамматики отражает тот факт, что замена нетерминального символа цепочкой не зависит от контекста, т.е. соседних символов.

Определение 1.3.2. Язык L называется контекстно-свободным, если существует КС-грамматика G, такая, что

$L = L(G)$.

Следующая теорема (теорема 2) позволяет во многих задачах ограничиваться рассмотрением только неукорачивающих КС-грамматик, т.е. таких, которые не содержат правил вида $A \to \varnothing$. Доказательству теоремы предпошлём следующую лемму.

Лемма 1.3.2. Пусть вывод $\varphi \overset{*}{\Rightarrow} \psi$ в КС-грамматике G состоит из применения множества правил

$F = \{P_1, P_2, \dots, P_k\}$ и $\varphi = \varphi_1, \varphi_2, \dots, \varphi_l$.

Тогда

$\psi = \psi_1 \psi_2 \dots \psi_l$, $\varphi_i \Rightarrow \psi_i, i = 1, \dots, l$

и наборы правил F_i для этих выводов удовлетворяют условию

$\bigcup F_i = F$.

Доказательство. Пусть $\varphi = \varphi^0 \Rightarrow \varphi^1 \Rightarrow \dots \Rightarrow \varphi^k = \psi$ в КС-грамматике $G = (V, T, P, S)$.

Рассмотрим

$G' = (V \cup \{/\}, T \cup \{/\}, P, S)$ и слово

$\varphi' = \varphi_1 | \varphi_2 | \varphi_3 | \dots | \varphi_l$.

Применим те же правила F к φ'.

Получим (легко доказывается индукцией по длине вывода), что

$\varphi' \overset{*}{\Rightarrow} \psi = \psi_1 | \psi_2 | \psi_3 | \dots | \psi_l$.

Вывод $\varphi_i \overset{*}{\Rightarrow} \psi_i$ делается применением набора

$F_i, i = 1, \dots, l$.

Значит $F = \bigcup F_i$. Ч.т.д.

Определение 1.3.3. Пусть $G = (V, T, P, S)$ - КС-грамматика.

Символ $A \in V - T$ называется *сводимым* к x, $x \in (V - T)^*$, если существует вывод $A \overset{*}{\Rightarrow} x$.

Теорема 1.2.4. (О неукорачивающих грамматиках). Пусть $G_1 = (V, T, P_1, S)$ есть КС-грамматика и множество P_2 состоит из всех правил вида $A \to \psi (\psi \neq \varnothing)$ таких, что в

P_1 найдётся правило $A \to \varphi$ и либо $\psi = \varphi$, либо ψ получается из φ выбрасыванием некоторых символов, сводимых к $\varnothing$.

Тогда КС-грамматика $G_2 = (V, T, P_2, S)$ - неукорачивающая и порождает язык $L_2 = L(G_2) = L(G_1) - \varnothing$.

Доказательство. Если $A \to \psi \in P_2$, то $A \overset{*}{\Rightarrow} \psi$ в G_1 и следовательно, если $S \overset{*}{\Rightarrow} x$ в G_2, то тотже вывод существует и в G_1, т.е. $L_1 \supseteq L_2$.

Обозначим теперь $G_0 = (V, T, P_1 \cup P_2, S) \cup L_0 = L(G_0)$. Очевидно, что $L_0 \supseteq L_1 \supseteq L_2$. Пусть $x \in L_0$, $x \notin L_2$ и $S \Rightarrow \varphi_1 \Rightarrow \varphi_2 \Rightarrow ... \Rightarrow \varphi_k = x$ - кратчайший вывод слова x в G_0. Возьмём первое применение правила вида $B \to \varnothing$. Пусть это будет в переходе $\varphi_j \Rightarrow \varphi_{j+1}$.

Найдём в выводе ближайшее слева к этому переходу место появления B.

Допустим, что это произошло в переходе

$\varphi_i \Rightarrow \varphi_{i+1}$. Пусть далее

$\varphi_i = \chi_1 A \chi_4$, $A \to \chi_2 B \chi_3$ и $\varphi_j = \psi_1 \psi_2 B \psi_3 \psi_4$.

Тогда

$\varphi_{i+1} = \chi_1 \chi_2 B \chi_3 \chi_4$,

$\varphi_{j+1} = \psi_1 \psi_2 \psi_3 \psi_4$,

$\chi_l \overset{*}{\Rightarrow} \psi_l, l = 1, 2, 3, 4$

(здесь мы пользуемся леммой 1.3.2).

Пусть последние выводы осуществляются, соответственно, за $\pi_l (l = 1, 2, 3, 4)$ шагов.

Значит φ_i сводится к φ_{i+1} в выводе $S \overset{*}{\Rightarrow} x$ за

$1 + \pi_l + \pi_2 + \pi_3 + \pi_4 + 1$ шагов.

В P_2 есть правило $A \to \chi_2 \chi_3$ (если $\chi_2 \chi_3 = \varnothing$, то прослеживаем назад A и проводим аналогичные рассуждения).

Следовательно, φ_i можно свести к φ_{j+1} за $1 + \pi_l + \pi_2 + \pi_3 + \pi_4$ шагов. Это противоречит тому, что рассматриваемый вывод $S \overset{*}{\Rightarrow} x$ кратчайший в G_0.

Поэтому $L_0 = L_2$ и $L_2 = L_1$. Ч.т.д.

Следующая теорема позволяет исключать из грамматики те нетерминальные символы, из которых выводится лишь конечное число слов языка.

Теорема 1.3.5. Для всякой КС-грамматики $G_1 = (V, T, P_1, S)$ существует эквивалентная ей грамматика $G_2 = (V, T, P_2, S)$, в которой $V_2 \subset V_1$ и из каждого $A \in V_2 - (T \cup S)$ выводимо бесконечное множество слов языка.

Доказательство. Грамматику G_2 построим следующим образом. Если правило $p = A \to \varphi \in P_1$ и $A, \varphi \in V_2^*$, то включаем p в P_2. Правила $p = A \to \varphi$, для которых $A \in V_2$ вообще выбрасываем. Если $p = A \to \varphi$, $A \in V_2$ и $\varphi \in V_2^*$, то включаем в P_2 все правила, которые получаются из P следующим образом: в цепочке $\varphi = \alpha_1 \alpha_2 \ldots \alpha_l$ заменяем те $\alpha_i, i = 1, 2, \ldots, l$, которые не входят в V_2, на какие-либо слова из T', выводимые из них.

Покажем, что $L_1 \supseteq L_2$. Действительно, пусть

$S \Rightarrow \varphi_1 \Rightarrow \varphi_2 \Rightarrow \ldots \Rightarrow \varphi_k = x$ в G_2.

Поскольку $\varphi_i \in V_2^*, i = 1, 2, \ldots, k$ и $V_2^* \subset V_1^*$,

то либо

$\varphi_i \Rightarrow \varphi_{i+1}$ в G_1,

либо

$\varphi_i \overset{*}{\Rightarrow} \varphi_{i+1}$ в G_1.

Значит $S \overset{*}{\Rightarrow} x$ в G_1 и $L_1 \supseteq L_2$.

Докажем обратное включение, $L_1 \subseteq L_2$. Для этого рассмотрим грамматику $G_3 = (V_1, T, P_1 \cup P_2, S)$ и, соответственно, язык $L_3 = L(G_3)$.

Возьмём самый короткий вывод в G_3 со следующими свойствами:

$\varphi \overset{*}{\Rightarrow} x, \varphi \in V_2^*, x \in T^*$

и x невыводимо из φ в G_2. Пусть этот вывод имеет вид:

$\varphi = \varphi_0 \Rightarrow \varphi_1 \Rightarrow \ldots \Rightarrow \varphi_k = x$

Тогда $\varphi_1 \in V_2^*$, иначе вывод $\varphi_1 \overset{*}{\Rightarrow} x$ был бы короче. Следовательно первое правило, применённое в рассматриваемом выводе $P_1 = A \to \psi (= \psi_1 \psi_2 \ldots \psi_l)$ и $P_1 \in P_2$.

Тогда

$\varphi_0 = \varphi' A' \varphi''_x$

$\varphi_1 = \varphi' \varphi_1 \varphi_2 \ldots \varphi_l \varphi''$,

$x = x' x_1 x_2 \ldots x_l x''$,

где

$\varphi' \overset{*}{\Rightarrow} x', \varphi'' \overset{*}{\Rightarrow} x'', \psi_i \overset{*}{\Rightarrow} x_i$ в G_3.

Заменив в правиле $A \to \psi$ те ψ_m, которых нет в V_2^* на x_m получаем правило вывода из P_2 и более короткий вывод цепочки x в G_3.

Значит $L_3 = L_2$ и $L_1 = L_2$. Ч.т.д.

1.4. Операции над КС-языками

В этом параграфе мы рассмотрим некоторые операции над КС-языками, которые сохраняют контекстно-свободную структуру языков. Основной результат формулируется в теореме 1.4.7.

Определение 1.4.6. Сопоставм каждому символу $a \in T$ алфавит T_α и множество $L_a \subseteq T_\alpha^*$.

Пусть $L_\varnothing = \varnothing$ и $L_{a_1,a_2,\ldots,a_r} = L_{a_1} L_{a_2} \ldots L_{a_r}$ для каждой цепочки $a_1 a_2 \ldots a_r \in T^*$.

Тогда отображение Re множества T^* во множество всех подмножеств множества $(\bigcup_\alpha T_\alpha)^*$ называется подстановкой.

Таким образом из каждого слова $x = a_1 \ldots a_k \in L$ в результате подстановок $a_i \to x_i \in L_{a_i}$ получаем слово

$z = x_1 \ldots x_k$.

Пусть $\tilde{L}\{z = x_1 \ldots x_k |\, x_i \in L_{a_i}; a_1 \ldots a_k \in L\}$.

Теорема 1.4.7. (о подстановке). Если языки L и L_{a_k} - КС-языки, то и язык $\tilde{L}$ - КС-язык.

Доказательство. Пусть L и $L_{a_k}, k = 1, 2, \ldots, n$ - КС-языки.

$L = L(G), G = (V, T, P, S)$,

$L_{a_k} = L(G_{a_k}), G_{a_k} = (V_{a_k}, T_{a_k}, P_{a_k}, S_{a_k})$

Сделаем замены нетерминальных символов так, чтобы

$(V_a - T_a) \cap (V_b - T_b) = \varnothing$ и $V \cap V_a = \varnothing$.

Построим грамматику $\tilde{G}$:

$$\tilde{G} = (V \cup (\bigcup_u V_{a_k}), \bigcup_k T_{a_k}, P \cup (\bigcup_k P_{a_k}) \cup \{a_k \to S_{a_k}\}, S)$$

Докажем, что язык $\tilde{L}$ совпадает с языком $L(\tilde{G})$.

1^0. Имеет место включение $\tilde{L} \subseteq L(\tilde{G})$.

Действительно, если $x \in \tilde{L}$, то z выводится из некоторого слова $x \in L$, которое имеет вид: $x = a_1 \dots a_3 \overset{*}{\Rightarrow} z$, но слово x выводится и в $L(\tilde{G})$, а тогда из x существует вывод $x \overset{*}{\Rightarrow} S_{a_1} \dots S_{a_3} \overset{*}{\Rightarrow} z$ в грамматике $\tilde{G}$.

2^0. Имеет место включение $L'(\tilde{G}) \subseteq \tilde{L}$.

Пусть $z \in (\bigcup_k T_{a_k})^*$. Так как $L(\tilde{G})$ - КС-язык и не существует в $\tilde{G}$ правил, таких что из символов $\bigcup V_{a_k}$ получаются символы из V, то существует вывод z такой, что до некоторого шага применяются только правила из P, а затем они не применяются, т.е. вывод в котором сначала применяются правила из ???, затем правила $a_k \to S_{a_k}$ и только потом остальные. В таком случае в момент t после окончания применения правил из P мы имеем слово из T^*, а так как алфавит не пересекается с $V \cup (V_a \breve{\ } T_a) \cap (V_b \breve{\ } Tb) = \varnothing$, тоиз каждого символа a_k выводится слово в алфавите V_{a_k}. Включение доказано.

Включения, доказанные в пунктах 1^0, 2^0 доказывают равенство $\tilde{L} = L(\tilde{G})$.

Следствие. Рассмотрим произвольные КС-языки L_1 и L_2. Поскольку множества $\{a, b\}$, $\{ab\}$ и $\{a\}^*$ (a и b - некоторые символы) являются КС-языками, то, согласно теореме 1.4.7, множества L_1 и L_2 L_1L_2 и L_1^* тоже являются КС-языками. Символом L_1L_2 обозначено множество цепочек вида xy, где $x \in L_1$, $y \in L_2$.

Для построения примеров, опровергающих те или иные утверждения, полезно иметь примеры языков, не являющихся контекстно-свободными. Следующая теорема даёт такой пример.

Теорема 1.4.8. Язык $L = \{a^n b^n c^n | n \geqslant 1\}$ не порождается никакой КС-грамматикой.

Доказательство. Пусть $L = L(G)$, где $G = (V, T, P, S)$ есть КС-грамматика. Можно считать, что из каждого нетерминального символа $A \in V - T$ выводится бесконечное множество слов языка.

Для каждого n зафиксируем последнее правило P_{i_n}, применённое при выводе $a^n b^n c^n$. Какое-то из правил встретится бесконечное число раз. Пусть это правило $P = A \to x$ и $|x| = m$. Возьмём n, такое, что $n > m$ и P применялось последним при выводе $a^n b^n c^n$.

Возможны такие случаи:

$xAb^k c^n \Rightarrow a^n b^n c^n$

и

$a^n b^k Ax \Rightarrow a^n b^n c^n, k \geqslant 1, |x| = 2n - k - m.$

Рассмотрим первый случай. Так как из A выводится бесконечно много слов, то существует вывод $A \overset{*}{\Rightarrow} y$, такой, что $|y| > m$. Тогда $|x| - |y| + k = 2n - k - m > 2n$, что невозможно.

Во втором случае рассуждаем аналогично. Ч.т.д.

Упражнение 4.1.9. Показать, что пересечение КС-языков не всегда является КС-языком.

2. Показать, что разность двух КС-языков не всегда является КС-языком.

3. Показать, что язык $L = \{a, b, c\}^* - \{a^n b^n c^n | n \geqslant 1\}$ есть КС-язык.

1.5. Неоднозначные грамматики и неоднозначные языки

Применительно к языкам программирования, порождающие грамматики доставляют средства для определения того, правильно или неправильно написана данная программа, т.е. является ли эта программа выводимой цепочкой в соответствующей грамматике. Более того, некоторые системы трансляции явно используют вывод данной цепочки для построения программы на другом языке. В связи с этим необходимо дать ответ на вопрос: когда можно считать два вывода одной и той же цепочки языка существенно различными? Для наших целей будет удобно отождествить все выводы некоторой цепочки в данной КС-грамматике, различающиеся лишь порядком применения правил. Среди всех таких выводов следующее определение выделяет вывод, который часто будет играть роль канонического вывода данной цепочки.

Определение 1.5.1. Пусть $G = (V, T, P, S)$ - есть КС-грамматика.

Вывод $S \Rightarrow \varphi_0 \Rightarrow \varphi_1 \Rightarrow \dots \Rightarrow \varphi_k$ является *левосторонним*, если для всех $i = 0, 1, \dots, k-1$ выполнено условие $\varphi_i = u_i A_i \psi_i, \varphi_{i+1} = u_i \chi_i \psi_i, A_i \to \chi_i \in P$ и $u_i \in T^*$.

Аналогично определяется *правосторонний* вывод.

Только что сформулированное определение приводит к весьма важному в практических приложениях понятию неоднозначности КС-грамматики.

Определение 1.5.2. КС-грамматика G называется *неоднозначной*, если в языке $L(G)$ найдётся по меньше мере одна цепочка, порождаемая двумя различными левосторонними выводами (из аксиомы S).

КС-грамматика, не являющаяся неоднозначной, называется *однозначной*.

Замечание. Рассматривая вывод некоторой цепочки в данной КС-грамматике G, часто используют понятие дерева вывода. Дерево вывода строится по выводу

$S \Rightarrow \alpha_1 \dots \alpha_{k_1} \Rightarrow \beta_1 \dots \beta_{k_2} \Rightarrow \dots \Rightarrow a_1 \dots a_k$

следующим образом: от вершины дерева (вершины уровня 1), помеченной символом S, отходит k_1 ветвей, заканчивающихся вершинами (вершинами второго уровня), помеченными слева направо символами $\alpha_1 \alpha_2 \dots \alpha_k$. Если в процессе вывода к некоторому нетерминальному символу α_i было применено правило $\alpha_i \to \gamma_1 \dots \gamma_m$, то α_i становится вершиной поддерева, от него отходит m ветвей, вершины, к которым они ведут, помечаются символами $\gamma_1 \dots \gamma_m$. Аналогичное построение проводится для вершин всех достигнутых уровней, помеченных нетерминальными символами. Вершина дерева называется максимальной, если от неё не отходит ветвей к вершинам более высокого уровня. Построение дерева вывода заканчивается, когда все максимальные вершины оказываются помеченными терминальными символами. Совокупность соответствующих меток (при осмотре максимальных вершин слева направо) образует терминальную цепочку $a_1 \dots a_l$ (пример дерева вывода см. на рис. стр. ???40).

Используя понятие дерева вывода можно сказать, что грамматика G является неоднозначной, если некоторая цепочка $w \in L(G)$ имеет два различных дерева вывода.

Определение 1.5.3. КС-язык L называется *существенно неоднозначным*, если каждая порождающая его КС-грамматика является неоднозначной. В противном случае язык L называется *однозначным*.

Примеры.

Пример 1.1. В АЛГОЛ-60 существуют следующие правила вывода.
< простая переменная> ::= < именователь переменной>
< именователь переменной> ::= < именователь>
< именователь массива> ::= < именователь>
< именователь процедуры > ::= < именователь>
< метка> ::= < именователь >
< именователь переключателя> ::= < именователь>

Добавив к этим правилам правила
$S \to$ ПП
$S \to$ ИМ
$S \to$ ИП
$S \to$ М
$S \to$ И Перекл
им $\to a$
(начальные символы АЛГОЛ-60 заменены здесь начальными буквами составляющих их слов), получим неоднозначную грамматику, для слова a в которой существует пять различных выводов. Источником неоднозначности являются правила с одинаковыми правыми частями.

Пример 1.2. Условный оператор в АЛГОЛ-60 может иметь вид
< условный оператор> ::= IF < логическое выражение> THEN <безусловный оператор>
либо вид
< условный оператор> ::= IF < логическое выражение>
< безусловный оператор> ::= ELSE < оператор>

С чем связано ограничение на то, что после THEN следует безусловный оператор?

Пусть после THEN может следовать оператор.

Тогда допустима конструкция

IF $x > 0$ THEN IF $x > 5$ THEN $x := x - 5$

ELSE $x := +5$; печать (x).

При этом возможны две её трактовки:

1) IF $x > 0$ THEN < оператор> ELSE $x := x + 5$,
где <оператор> есть IF $x > 5$ THEN $x := x - 5$

2) IF $x > 0$ THEN < оператор>, где <оператор> есть
IF $x > 5$ THEN $x := x - 5$ ELSE $x := +5$;

При $x = -1$ первая трактовка приведёт к печати значения $x = 4$, а вторая - $x = -1$.

При $x = 1$ первая трактовка даёт результат $x = 1$, вторая - $x = 6$.

Отмеченная неоднозначность требует применения конструкции, используемой в АЛГОЛ-60.

`Пример 1.3.` Пусть $G_1 = (V_1, T, P, S_1)$ есть КС-грамматика, причём $T = \{a\}$, $P_1 = \{S \to SS, S \to a^k b^k\}$, и
$G_2 = (V_2, T, P, S_2)$ - другая КС-грамматика, с правилами вывода $P_2 = \{S \to a^k b^k S, S \to a^k b^k\}$.

Тогда $L(G_1) = L(G_2) = \{(a^k b^k)^n, n \geqslant 1\}$
при этом грамматика G_2 - однозначная, а G_1 - неоднозначная.

`Пример 1.4.` Язык $\{a^n b^n c^m\} \cup \{a^n b^m c^m\}$, $m, n \leqslant 1$ является существенно неоднозначным – любая порождающая его грамматика неоднозначна.

Объединение однозначных КС-языков не обязательно является однозначным КС-языком, однако, если эти языки не пересекаются, то их объединение является КС-языком.

<u>*Теорема 1.5.3.*</u> Если однозначные КС-языки $L_1, \ldots, L_r$ попарно не пересекаются, то язык $L = \bigcup_{i=1}^{r} L_i$ является однозначным.

Доказательство. Не ограничивая общности, можно считать, что $r = 2$.

Пусть
$G_1 = (V_1, T, P, S_1)$,
$G_2 = (V_2, T, P, S_2)$ и
$L_i = L(G_i)$, $i = 1, 2$ - однозначные КС-языки.

Пусть также $(V_1 - T) \cap (V_2 - T) = \varnothing$ (этого всегда можно добиться переобозначением символов вспомогательных алфавитов).

Рассмотрим КС-грамматику

$G_1 = (V_1, T, P, S_1)$, где

$S \notin V_1 \cup V_2$, $V = V_1 \cup V_2 \cup S$ и

$P = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$.

Очевидно, что

$L(G) = L_1 \cup L_2$.

Пусть
$S \Rightarrow \varphi_1 \Rightarrow \ldots \Rightarrow \varphi_k = x$ и
$S \Rightarrow \psi_1 \Rightarrow \ldots \Rightarrow \psi_l = x$ -
левосторонние выводы цепочки $x \in L(G)$.
Так как P содержит лишь два правила вида $S \to \chi$,
то $\varphi_i \in \{S_1, S_2\}$ и $\psi_i \in \{S_1, S_2\}$.
Предположим, что $\varphi_1 = \psi_1 = S_1$. Из соотношения

$$(V_1 - T) \cap (V_2 - T) = \varnothing$$

следует, что в обоих выводах применяются лишь правила P_1, и, поскольку грамматика G_1 является однозначной, данные выводы совпадают.

Аналогично доказывается, что выводы совпадают в случае $\varphi_i = \psi_i = S_2$. Предположим, что $\varphi_i = S_1$ и $\psi_1 = S_2$. Тогда первый вывод (т.е. $S \Rightarrow \varphi_1 \Rightarrow \ldots \Rightarrow \varphi_k = x$ является выводом в грамматике G_1, а второй – выводом в грамматике G_2. Значит $x \in L_1 \cap L_2$, что противоречит условию $L_1 \cap L_2 = \varnothing$. Таким образом, оба вывода слова x совпадают и в этом случае. Грамматика G - однозначная. Ч.т.д.

1.6. Магазинные автоматы

В этом параграфе мы рассматриваем класс устройств, называемых *автоматами с магазинной памятью*, каждый из которых обладает тем свойством, что прочитав слово, принадлежащее данному КС-языку, он некоторым определённым образом отмечает этот факт. Во многих теоретических и прикладных вопросах, можно с успехом заменить рассмотрение данного КС-языка изучением свойств соответствующего автомата с магазинной памятью.

Дано сначала наглядное неформальное описание автомата с магазинной памятью (МП-автомата). Такой автомат (рис ?) можно представлять себе состоящим из

а) входной ленты, на которой в начальный момент времени находится входная цепочка (слово в некотором алфавите T),

б) читающего устройства, которое может находится в конечном числе состояний $p, q, r, \ldots$

в) ленты магазина (ленты магазинной памяти), на которой в начальный момент находится символ z_0 некоторого конечного алфавита Z.

Функционирование автомата во времени происходит следующим образом (см. рис. 1.1.):

Либо 1) входная лента вдвигается на одну позицию в читающее устройство и в зависимости

от последнего символа z, записанного на ленте магазина,

состояния читающего устройства p

и считанного с входной ленты символа a

автомат записывает на магазинную ленту, начиная с той позиции, где находится символ z, некоторую цепочку ω в алфавите Z и переходит в новое состояние q.

Либо 2) считывания с входной ленты не проходит, а в зависимости от последнего символа магазина и состояния автомат записывает в магазин цепочку ω и переходит в новое состояние q.

Отметим, что переходы обоих видов могут совершаться автоматом *недетерминированно*, т.е. ситуации на предшествующем такте может соответствовать конечное число пар вида (ω, q) - цепочки, которая будет напечатана к следующему моменту времени на ленте магазина, и состояния.

Если в итоге работы получается положение, когда с входной ленты считаны все символы входной цепочки и выполнено некоторое условие остановки автомата, то говорят, что автомат *допустил* входную цепочку.

В качестве *условий остановки* мы будем брать следующие:

а) автомат находится в одном из *отмеченных* (заранее) состояний (мы в этом случае говорим о F-автомате), б) магазин автомата пуст (N-автомат).

Дадим теперь формальное определение автомата с магазинной памятью.

Определение 1.6.1. N-автоматом (автоматом типа N) называется упорядоченная шестёрка

$N \leftrightarrow N = (Z, K, T, \delta, z_0, q_0)$,

где

$Z = \{A, B, C, \dots, D\}$ - конечное множество символов (алфавит магазина),

$K = \{p, q, \dots, r\}$ - конечное множество (множество состояний автомата),

$T = \{a, b, \dots, c\}$ - конечное множество (входной алфавит автомата),

δ - заданное отображение множества $Z \times K \times (T \cup \varnothing$ на множество всех подмножеств множества $Z^* \times K$ (функция переходов автомата)

$z_0 \in Z$ - начальный символ в магазине (маркер магазина)

$q_0 \in K$ - начальное состояние автомата.

F-автоматом (автоматом типа F) называется упорядоченная семёрка

$F = (Z, K, T, \delta, z_0, q_0, F)$,

где $Z, K, T, \delta, z_0, q_0$ имеют тот же смысл что и для N-автомата, а $F \subseteq K$ - множество отмеченных (заключительных) состояний.

Введём некоторые обозначения.

Пусть в данный момент

- ▷ в магазине автомата находится слово vz,
- ▷ автомат находится в состоянии p,
- ▷ на входе автомата находится слово ax.

Набор (vz, p, ax) будем называть *ситуацией*, в которой находится автомат.

Пусть функция перехода автомата δ такова, что $\delta(z, p, a) \ni (\omega, q)$.

Совокупность этих условий будем обозначать

$(vx, p, ax) \vdash (vw, q, x)$.

Если

$(w_1, p_1, x_1) \vdash (w_2, p_2, x_2) \vdash ... \vdash (w_n, p_n, x_n)$,
то будем писать

$(w_1, p_1, x_1) \overset{*}{\vdash} (w_n, p_n, x_n)$.

Иногда значки $\vdash$ и $\overset{*}{\vdash}$ мы будем употреблять для обозначения фактически совершённых автоматом переходов.

Определение 1.6.2. Множество слов $L(N)$ будем называть множеством слов, допускаемых N-автоматом, если

$L(N) = \{x | x \in T^*, (z_0, q_o, x) \overset{*}{\vdash} (\varnothing, q, \varnothing), q \in K\}$.

Множество

$L(F) = \{x | x \in T^*, (z_0, q_o, x) \overset{*}{\vdash} (\omega, P, \varnothing), \omega \in Z^*\}$
называется множеством слов, допускаемых F-автоматом.

Цель двух нижеследующих теорем показать, что классы языков, допускаемых N- и F-автоматами, совпадают.

Теорема 1.6.3. (FN-теорема). Для каждого автомата типа F существует автомат типа N, такой, что $L(F) = L(N)$.

Доказательство. Пусть дан F-автомат

$F = (Z, K, T, \delta, z_0, q_0, F)$.

Возьмём такой N-автомат

$N = (Z \cup \bar{z}_0, K \cup \bar{q}_o \cup \bar{q}_1, T, \bar{\delta}, \bar{z}_0, \bar{q}_0)$,
где

$\bar{q}_o, \bar{q}_1 \notin K, \bar{z}_0 \notin Z$
и функция переходов определена следующими равенствами:

1) $\bar{\delta}(\bar{z}_0, \bar{q}_0, \varnothing) = (\bar{z}_0, z_0, q_0)$,
2) $\bar{\delta}(z, q, a) = \delta(z, q, a)$ для всех $z \in Z, q \in K$,
3) $\bar{\delta}(z, f, \varnothing) = (z, \bar{q}_1)$ для $f \in F, z \in Z \cup \bar{z}_o$,
4) $\bar{\delta}(z, \bar{q}_1, \varnothing) = (\varnothing, \bar{q}_1)$ для $z \in Z \cup \bar{z}$

Пусть $x \in L(F)$, т.е.

$(z_0, q_0, x) \overset{*}{\vdash} (w, f, \varnothing)$.

Тогда

$$(\bar{z}_0, \bar{q}_0, x) \vdash (\bar{z}_0, q_0, x) \overset{*}{\vdash} (\bar{z}_0 w, f, \varnothing) \vdash (\bar{z}_0 w, \bar{q}_1, \varnothing) \overset{*}{\vdash} (\varnothing, \bar{q}_1, \varnothing),$$

т.е. $x \in L(N)$ и $L(F) \subseteq L(N)$.

Пусть $x \in L(N)$. В F-автомате первый переход однозначен $(\bar{z}_0, \bar{q}_0, x) \vdash (\bar{z}_0, q_0, x)$. Далее рассуждаем так: $\bar{z}_0$ может быть стёрто (в автомате N) лишь в состоянии $\bar{q}_1$ (см. выше п. 4) описания функции $\bar{\delta}$. В состояние $\bar{q}_1$ можно перейти только из состояния f.

Таким образом,

$$(\bar{z}_0 z_0, q_0, x) \overset{*}{\vdash} (\bar{z}_0 w, f, \varnothing) \vdash (\bar{z}_0 w, \bar{q}_1, \varnothing) \overset{*}{\vdash} (\varnothing, \bar{q}_1, \varnothing).$$

Поскольку при переходе из состояния f в состояние $\bar{q}_1$ и при всех переходах, совершаемых в состоянии $\bar{q}_1$, с входной ленты ничего не считывается (см. п.п. 3 и 4 опсания $\bar{\delta}$), то

$$(z_0, q_0, x) \overset{*}{\vdash} (w, f, \varnothing),$$

т.е. $x \in L(F)$ и $L(N) \subseteq L(F)$. Ч.т.д.

Теорема 1.6.4. (NF-теорема). Для каждого автомата типа N существует автомат типа F, такой, что $L(N) = L(F)$.

Доказательство. Пусть дан автомат

$N = (Z, K, T, \delta, z_0, f_0)$.

Возьмём F-автомат

$F = (Z \cup \bar{z}_0, K \cup \bar{f} \cup \bar{q}_0, T, \bar{\delta}, \bar{z}_0, \bar{q}_0, f)$,

где

$f \notin K, \bar{z}_0 \notin Z$,

а функция переходов $\bar{\delta}$ определяется следующими равенствами:

1) $\bar{\delta}(\bar{z}_0, \bar{q}_0, \varnothing) = (\bar{z}_0, z_0, q_0)$,
2) $\bar{\delta}(z, q, a) = \delta(z, q, a)$ для всех $z \in Z, q \in K, a \in T \cup \varnothing$,
3) $\bar{\delta}(\bar{z}_0, q, \varnothing) = (\varnothing, f)$ для $q \in K$.

Пусть слово $x \in L(N)$, т.е. $(z_0, q_0, x_0) \overset{*}{\vdash} (\varnothing, q, \varnothing)$.

Тогда

$$(\bar{z}_0, \bar{q}_0, x) \vdash (\bar{z}_0 z_0, q_0, x) \overset{*}{\vdash} (\bar{z}_0, q_0, \varnothing) \vdash (\varnothing, f, \varnothing).$$

Значит $x \in L(F)$ и $L(N) \subseteq L(F)$.

Пусть теперь $x \in L(F)$, т.е. $(\bar{z}_0, \bar{q}_0, x) \overset{*}{\vdash} (w, f, \varnothing)$. Проследим этот переход пэтапно в обратную сторону. В состояние f можно перейти, только используя правило п. 3 описания функции $\bar{\delta}$, т.е.

$(\bar{z}_0, \bar{q}_0, x) \overset{*}{\vdash} (w\bar{z}_0, q, \varnothing) \vdash (w, f, \varnothing), q \in K.$

Так как символ $\bar{z}_0$ никогда в магазин не пишется, то должно выполняться равенство $w = \varnothing$. Значит

$(\bar{z}_0, \bar{q}_0, x) \overset{*}{\vdash} (\bar{z}_0, q_0, \varnothing).$

Первый переход в F-автомате однозначен (см. п. 1 описания функции $\bar{\delta}$).

Поэтому выполнены соотношения:

$(\bar{z}_0, \bar{q}_0, x) \vdash (\bar{z}_0 z_0, \bar{q}_0, x) \overset{*}{\vdash} (\bar{z}_0, q, \varnothing).$

Те же шаги(см. п.2 описания $\bar{\delta}$) для N-автомата дают

$(z_0, q_0, x) \overset{*}{\vdash} (\varnothing, q, \varnothing),$

т.е.

$x \in L(N)$ и $L(F) \subseteq L(N)$

Ч.т.д.

Сейчас мы приступаем к доказательству двух теорем, содержащих основное в этом параграфе утверждение об эквивалентности классов языков, порождаемых КС-грамматиками и допускаемых автоматами с магазинной памятью.

Теорема 1.6.5. (GN-теорема). Для любой КС-грамматики G существует N-автомат такой, что $L(G) = L(N)$.

Доказательство. Пусть $G = (V, T, P, S)$. Определим автомат $N = (V, \{q\}, T, \delta, q, S)$, где функция переходов δ удовлетворяет следующим равенствам:

1) $\delta(a, q, a) = (\varnothing, q)$ для всех $a \in T$

2) $\delta(A, q, \varnothing) = \{(\varphi^R, q)\}$, для всех правил $A \to \varphi \in P$ [1]

Пусть слово $x \in L(G)$. Возьмём левосторонний вывод x

$S \Rightarrow x_1 A_1 \varphi_1 \Rightarrow x_1 x_2 A_2 \varphi_2 \Rightarrow \ldots \Rightarrow x_1 x_2 \ldots A_{n-1} \varphi_{n-1} \Rightarrow$
$\Rightarrow x_1 x_2 \ldots x_n = x$

[1] Символом φ^R обозначается цепочка, состоящая из тех же символов, что и цепочка φ, но написанных в обратном порядке.

Тогда

$(S, q, x) \vdash (\varphi_1^R A_1 x_1^R, q, x) \overset{*}{\vdash} (\varphi_1^R A_1, q, x_1 x_2 \dots x_n) \vdash (\varphi_2^R A_2, q, x_2 \dots x_n) \overset{*}{\vdash} (\varphi_2^R A_2, q, x_3 \dots x_n) \vdash (\varphi_{n-1}^R A_{n-1}, q, x_n \dots x_n) \vdash (x_n^R, q, x_n) \overset{*}{\vdash} (\varnothing, q, \varnothing)$,

т.е. $x \in L(N)$ и $L(G) \subseteq L(N)$

Пусть теперь $x \in L(N)$, т.е. $(S, q, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$. Пусть $x = a_1 a_2 \dots a_{n-1} a_n$. Обозначим через $\psi_1, \psi_2, \dots, \psi_n$ состояния магазина до, а через $\varphi_1, \varphi_2, \dots \varphi_n$ после того, как будет согласно определению функции перехода δ из (1.6.1) считано, собственно, $a_1 a_2 \dots a_{n-1} a_n$. Используя эти обозначения, можно записать следующую последовательность переходов:

$(S, q, x) \overset{*}{\vdash} (\psi_1, q, a_1 a_2 \dots a_n) \vdash (\varphi_1, q, a_2 \dots a_n) \overset{*}{\vdash} (\psi_2, q, a_2 \dots a_n) \vdash (\varphi_2, q, a_3 \dots a_n) \overset{*}{\vdash} \dots \overset{*}{\vdash} (\psi_{n-1}, q, a_{n-1} \dots a_n) \vdash (\varphi_n, q, a_n) \overset{*}{\vdash} (\psi_n, q, a_n) \vdash (\varphi_n, q, \varnothing) \overset{*}{\vdash} (\varnothing, q, \varnothing)$

Это значит, что

1) $\psi_i = \varphi_i a_i$ для $i = 1, 2, \dots, n$ и

2) $S \overset{*}{\Rightarrow} \psi_i^R, \varphi_i^R \Rightarrow \psi_{i+1}^R$ для $i = 1, 2, \dots, n$ и $\varphi_n^R \overset{*}{\Rightarrow} \varnothing$

Объединяя всё, получаем вывод цепочки x в грамматике G

$S \overset{*}{\Rightarrow} \psi_i^R = a_1 \varphi_1^R \overset{*}{\Rightarrow} a_1 \psi_2^R = a_1 a_2 \varphi_2^R \overset{*}{\Rightarrow}$

$\overset{*}{\Rightarrow} a_1 \dots a_{n-1} \psi_n^R = a_1 \dots a_{n-1} \varphi_n^R \overset{*}{\Rightarrow} a_1 \dots a_n = x$

Значит

$x \in L(G)$ и $L(G) = L(N)$, ч.т.д.

Теорема 1.6.6. (NG-теорема). Для любого N-автомата N существует КС-грамматика G, порождающая язык, допускаемый автоматом N.

Доказательство. Пусть $N = (Z, K, T, \delta, z_0, q_0)$

Построим КС-грамматику $G = (V, T, P, S)$ следующим образом.

Алфавит грамматики $V = \{A_{zpq} \cup T \cup S, p, q \in K, z \in Z$.

Смысл A_{zpq} следующий: слова $x \in T^*$, выводимые из нетерминального символа A_{zpq} в грамматике G (т.е. $A_{zpq} \overset{*}{\Rightarrow} x$), это те слова, которые переводят автомат N из состояния

p в состояние q так, что при этом стирается записанный в магазин последний (до начала перехода) символ z, т.к. Эти цепочки обладают свойством

$(z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$.

Множество P правил вывода грамматики G построим из следующих правил:

1) $S \to A_{z_0 p_0 q_i}$ для всех $p_i \in K$,

2) $A_{zpq} \to a$, если $\delta(z, p, a) \ni (\varnothing, q)$, т.е. $(z, p, a) \vdash (\varnothing, q, \varnothing)$

3) Если $\delta(z, p, a) \ni (z_1 z_2 \dots z_k, r)$, т.е.

$(z, p, a) \vdash (z_1 z_2 \dots z_k, r, \varnothing)$,

то для каждой последовательности состояний

$r_k, r_{k-1}, \dots, r_0$ такой, что $r_k = r$ и

$(z_k, r_k, y_k) \overset{*}{\vdash} (\varnothing, r_{k-1}, \varnothing)$,

$(z_{k-1}, r_{k-1}, y_{k-1}) \overset{*}{\vdash} (\varnothing, r_{k-2}, \varnothing)$,

...

$(z_2, r_2, y_2) \overset{*}{\vdash} (\varnothing, r_1, \varnothing)$,

$(z_1, r_1, y_1) \overset{*}{\vdash} (\varnothing, r_0, \varnothing)$

для подходящих цепочек $y_i \in T^*, i = 1, 2, \dots, K$ в множество P добавляется правило

$A_{zpr_0} \to aA_{z_k r r_{k-1}} A_{z_{k-1} r_{k-1} r_{k-2}} \dots A_{z_2 r_2 r_1} A_{z_1 r_1 r_0}$

Можно включить в P «бесполезные» правила, т.е. выписать правила вида

$A_{zpr_0} \to aA_{z_k r_k r_{k-1}} A_{z_{k-1} r_{k-1} r_{k-2}} \dots A_{z_2 r_2 r_1} A_{z_1 r_1 r_0}$

для каждой последовательности $r_k, r_{k-1}, \dots, r_1 r_0$ из $k + 1$ состояний).

А) Докажем, что $L(N) \subseteq L(G)$, т.е. что из условия

$(z_0, q_0, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$

следует, что существует вывод

$S \Rightarrow A_{z_0 q_0 q} \overset{*}{\Rightarrow} x$.

Первый шаг вывода * - это применение правила вида 1).

Для того, чтобы доказать существование остальной части вывода, докажем более общее утверждение:

если $(z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$, то

$A_{zpq} \overset{*}{\Rightarrow} x$
(для любых P и z, а не только для q_0, z_0).

Доказательство этого утверждения проведём индукцией по числу n тактов работы автомата.

Если $n = 1$, то $x = a$

$(z, p, a) \vdash (\varnothing, q, \varnothing)$

Но тогда $A_{zpq} \to a$ есть правило (типа 2) из P. Пусть наше утверждение доказано для всех случаев, когда число тактов работы автомата меньше n. Докажем его для случая, когда число тактов равно n.

Первый такт имеет видов

$(z, p, ax') \vdash (z_1 \dots z_k, r, x')$.

Зафиксируем моменты $\tau_k, \tau_{k-1}, \tau_1$, когда будут исчезать из магазина последовательно символы $z_k, z_{k-1}, \dots, z_1$. Пусть автомат при этом будет переходить в состояния

$r_{k-1}, r_{k-2}, \dots, r_1, r_0 = q$.

Положим $r_k = r$. Обозначим через $y_k, y_{k-1}, \dots, y_1$ те части слова x', которые считываются автоматом, соответственно, к моменту τ, между моментами τ_k и τ_{k-1} и так далее, до интервала $[\tau_2, \tau_1]$. Используя эти обозначения можно записать равенство

$x' = y_k, y_{k-1}, \dots, y_2, y_1$,
а также следующим образом описать ситуации в моменты времени

$0, \tau_k, \tau_{k-1}, \tau_2, \tau_1$:

в начале работы	$(z_1, \dots, z_k,$	$r,$	$y_k, y_{k-1}, \dots, y_2, y_1)$
в момент τ_k	$(z_1, \dots, z_{k-1},$	$r_{k-1},$	$y_{k-1}, \dots, y_2, y_1)$
в момент τ_{k-1}	$(z_1, \dots, z_{k-2},$	$r_{k-2},$	$y_{k-2}, \dots, y_2, y_1)$
в момент τ_2	$(z_1,$	$r_1,$	$y_1)$
в момент τ_1	$(\varnothing,$	$r_0,$	$\varnothing)$

Таким образом, справедливы соотношения

$(z, p, ay_k \dots y_1) \vdash$

$(z_1 \dots z_k, r_k, y_k \dots y_1) \overset{*}{\vdash}$

$(z_1 \dots z_{k-1}, r_{k-1}, y_{k-1} \dots y_1) \overset{*}{\vdash}$

$(z_1, r1, y_1) \overset{*}{\vdash}$
$(\varnothing, r_0, \varnothing)$

Каждый из переходов от состояния r_i к состоянию $r_{i-1}, i = k, k-1, \dots, 1$ эквивалентен такому

$(z_i, r_i, y_i) \overset{*}{\vdash} (\varnothing, r_{i-1}, \varnothing)$

Число тактов работы автомата в этом переходе меньше n, поэтому существует вывод в грамматике G

$A_{z_i r_i r_{i-1}} \overset{*}{\Rightarrow} y_i \qquad (*)$

Переход автомата от состояния p в состояние r_k эквивалентен

$(z, p, a) \vdash (z_1 \dots z_k, r_k, \varnothing)$.

Применяя правило вывода типа 3, получаем

$A_{zpr_0} \to aA_{z_k r_k r_{k-1}} A_{z_{k-1} r_{k-1} r_{k-2}} \dots A_{z_2 r_2 r_1} A_{z_1 r_1 r_0}$

Вместе с соотношениями (*) это показывает существование вывода

$A_{zpq} \overset{*}{\Rightarrow} a y_k y_{k-1} \dots y_2 y_1 = x$

Пункт A) доказан.

B) Докажем, что $L(G) \subseteq L(N)$ Сначала докажем вспомогательное утверждение, обратное тому, которое было показано в пункте A):

если $A_{zpq} \overset{*}{\Rightarrow} x$, то $(z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$.

Доказательство этого утверждения проведём индукцией по длине n вывода x.

Если $n = 1$, то $x = a$ и $A_{zpq} \to a$. Следовательно, возможен переход

$(z, p, a) \vdash (\varnothing, q, \varnothing)$.

Пусть наше утверждение доказано для выводов с длинами, меньше n. Докажем его для случая, когда вывод цепочки совершается за n шагов.

Первый шаг имеет вид

$A_{zpq} \to aA_{z_k r_k r_{k-1}} \dots A_{z_2 r_2 r_1} A_{z_1 r_1 q}$

Это соответствует разбиению x на подслова

$x = a x_k x_{k-1} \dots x_1$

и переходу автомата $(z, p, a) \vdash (z_1 z_2 \dots z_k, r_k, \varnothing)$, причём существуют выводы $A_{z_i r_i r_{i-1}} \overset{*}{\Rightarrow} x_i, i = 1, 2, \dots, k$ длины меньшей n.

Но тогда существуют переходы автомата

$(z_i, r_i, x_i) \overset{*}{\vdash} (\varnothing, r_{i-1}, \varnothing), i = 1, 2, \dots, k.$

Значит существует и нужный нам переход автомата

$(z, p, x) = (z, p, ax_k \dots x_1) \vdash (z_1 \dots z_k, r_k, x_k \dots x_1) \overset{*}{\vdash} (z_1 \dots z_{k-1}, r_{k-1}, x_{k-1} \dots x_1) \overset{*}{\vdash} \dots \overset{*}{\vdash} (z_1 z_2, r_2, x_2 x_1) \overset{*}{\vdash} (z_1, r_1, x_1) \overset{*}{\vdash} (\varnothing, q, \varnothing)$

Пусть теперь $x \in L(G)$ и $S \Rightarrow \varphi_1 \Rightarrow \dots \Rightarrow \varphi_k = x$ его вывод.

Поскольку первый шаг этого вывода состоит в применении правила вида 1), то $\varphi_1 = A_{z_0 q_0 p}$.

Так как существует вывод $A_{z_0 q_0 p} \overset{*}{\Rightarrow} x$, то существует (по доказанному выше) переход автомата

$(z_0, q_0, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$, т.е. $x \in L(N)$.

Теорема доказана полностью.

Глава 2.

Синтаксический анализ

2.1. Проблема принадлежности слова языку

При заданной порождающей грамматике G можно произвольно порождать слова языка $L(G)$ и деревья их выводов, исходя из начального символа S. С другой стороны, например, трансляторы выполняют противоположную задачу:

1) по заданному слову x в грамматике G определить принадлежит ли это слово языку $L(G)$.

2) восстановить вывод этого слова в случае положительного ответа на первый вопрос. Поставленную задачу будем называть *задачей синтаксического анализа*.

Формально, для решения задачи синтаксического анализа в случае КС-языка L, порождаемого КС-грамматикой G, можно использовать эквивалентность КС-языков и магазинных автоматов: по заданной грамматике G построить эквивалентный N-автомат, допускающий лишь слова, порождаемые грамматикой G. Неэффективность такого подхода очевидна – построенный по GN-теореме автомат будет недетерминированным, требующим возвратов в работе. Поэтому мы ограничимся лишь такими процессами решения задачи синтаксического анализа, которые осуществляются некоторым детерминированным устройством согласно заданной конечной системе чётких предписаний. Такое интуитивное понятие процесса решения формализуется в математической логике с помощью понятия алгоритма. Общего алгоритма решения поставленной задачи нет, но для языков, порождаемых неукорачивающими грамматикам, это сделать можно. Ниже также будет показано, что проблема принадлежности слова x КС-языку $L(G)$ алгоритмически разрешима – иными словами, существует алгоритм, позволяющий определить принадлежит ли заданное произвольное слово заданному произвольному КС-языку.

Теорема 2.1.1. Пусть задана произвольная неукорачивающая грамматика $G(V,T,P,S)$, порождаемый ею язык $L(G)$ и слово x длины $l \geqslant 1$ в алфавите T. Проблема принадлежности слова x языку $L(G)$ алгоритмически разрешима.

Доказательство теоремы основано на возможности ограничиться просмотром всех выводов конечной длины, зависящей от длины слова x. Чтобы сделать это, нужно из всех выводов исключить цикл типа:

$S \to \alpha \to \underbrace{\beta \to \gamma \to \alpha} \to abc.$

В противном случае для любого слова с возможностью цикла при выводе можно получить вывод сколь угодно длинный. Легко видеть, что для произвольного вывода в любой грамматике существует равносильный ему (т.е. имеющий те же первую и последнюю цепочки) бесповторный вывод, в котором никакая цепочка не встречается более одного раза.

Доказательство теоремы. Пусть алфавит V содержит m символов, множество P состоит из n правил подстановки. Рассмотрим произвольный вывод слова $y \in T^*$ длины l :

$$S \to \varphi_1 \stackrel{*}{\Rightarrow} \psi_1 \to \varphi_2 \stackrel{*}{\Rightarrow} \psi_2 \to \varphi_3 \stackrel{*}{\Rightarrow} ... \to \varphi_l \stackrel{*}{\Rightarrow} \psi_l = y,$$

в котором $\varphi_i \psi_i$ - цепочки длины i, а бесповторные выводы

$\overset{*}{\Rightarrow}$ не увеличивают длину цепочки.

Рассмотрим бесповторный вывод $\overset{*}{\Rightarrow}$; так как грамматика неукорачивающая, а $|\varphi_i| = |\psi_i|$ по построению вывода, следовательно, в выводе применяются лишь правила вида $\varphi \to \psi$, у которых $|\varphi| = |\psi|$.

Правила $\alpha \to \alpha_1, \alpha, \alpha_1 \in V$ порождают не более $(n+1)^i$ различных цепочек, правила $\alpha\beta \to \alpha_1\beta_1; \alpha, \beta, \alpha_1, \beta_1 \in V$, порождают не более $(n+1)^{i-1}$ различных цепочек, поэтому длина вывода (количество подстановок в выводе) $\varphi_i \overset{*}{\Rightarrow} \psi_i$ не превышает

$$(n+1)^i + (n+1)^{i-1} + ... + 1 < (n+1)^{i+1}.$$

Стрелок без знака $*$, увеличивающих длину слова на единицу, будет не больше $l-1$. Таким образом общая длина вывода слова y не больше

$$N = l - 1 + \sum_{i=1}^{l} (n+1)^{i+1}.$$

Имея слово x длины l нужно просмотреть все бесповторные выводы, длина которых не больше N.

Если среди порождённых такими выводами слов есть слово x, то $x \in L(G)$ и одновременно получен вывод этого слова. Нами доказано существование алгоритма решения задачи синтаксического анализа для неукорачивающей грамматики G.

Теорема 2.1.2. Существует алгоритм, позволяющий узнать, принадлежит ли данное слово x произвольному данному КС-языку.

Доказательство. Пусть $x \in T^*$, $L = L(G)$, где G - некоторая КС-грамматика $G = G(V, T, P, S)$. Если $x = \varnothing$, то алгоритм, решающий вопрос о принадлежности $x \in L(G)$ состоит в следующем. Рассмотрим последовательность множеств w_i, определяемых реккурентно:

$w_1 = \{A | A \to \varnothing, A \in V\}$

$\vdots$

$w_{k+1} = \{A|A \to \varphi, \varphi \in w_k^*\} \cup \{w_k\}$,

$k = 1, 2, \ldots, n \leqslant m - 1$,

где m - число нетерминальных символов.

Нетрудно показать, что $w_k \subseteq w_{k+1}$ для всех k и $\varnothing \in L(G)$ тогда и только тогда, когда $S \in w_n$.

Пусть $x \neq \varnothing$. Тогда по грамматике G можно эффективно построить неукорачивающую КС-грамматику $G' = (V, T, P', S)$ такую, что $L(G') = L(G) - \varnothing$. Для грамматики G' задача синтаксического анализа алгоритмически разрешима. Грамматику G' легко построить, используя описанное множество w_n: каждое правило $A \to \varphi, \varphi \neq \varnothing$ исходной грамматики G порождает совокупность правил $A \to \varphi'$ грамматики G', таких что φ' получено из φ удалением всевозможных вхождений символов из w_n. Доказательство теоремы закончено.

2.2. Анализ сверху-вниз и снизу-вверх

Существуют две разные стратегии синтаксического анализа, называемые анализом снизу-вверх (восходящий) и анализом сверху-вниз. Будем в дальнейшем рассматривать левосторонний анализ, закрепив общий порядок обработки символов в цепочке – слева направо.

Анализ сверху-вниз называется так по способу построения дерева вывода. На каждом этапе анализа в дереве создаются некоторые дуги (быть может, неправильные), при этом двигаются вниз по дереву от вершины, помеченной начальным символом, к слову из терминальных символов, расположенному снизу. Первым шагом является проверка того, нельзя ли анализируемую цепочку привести к виду правой части $\alpha_1\alpha_2 \ldots \alpha_n$ некоторого правила $S \to \alpha_1\alpha_2 \ldots \alpha_n$.

Это делается следующим образом: чтобы вывод был правильным в том случае, если α_1 - терминальный символ, цепочка должна начинаться этим терминальным символом.

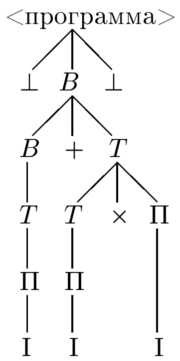
Если α_1 - нетерминальный символ, ставится и проверяется подцель – посмотреть, нельзя ли начало некоторой цепочки привести к α_1. Если это оказывается возможным, то α_2 проверяется таким же образом, затем α_3 и т.д. Если не удастся найти ничего подходящего для некоторого α_i, то пробуют применить другое порождающее правило $S \to \alpha_1'\alpha_2' \dots \alpha_m'$.

Подцели A проверяются также: пробуют применить порождающее правило $A \to \alpha_1\alpha_2 \dots \alpha_n$. Так постоянно генерируются и испытываются новые подцели. Если не находят новой подцели, то передаётся сообщение о неудаче на следующий, более высокий уровень, где проверяется другая альтернатива.

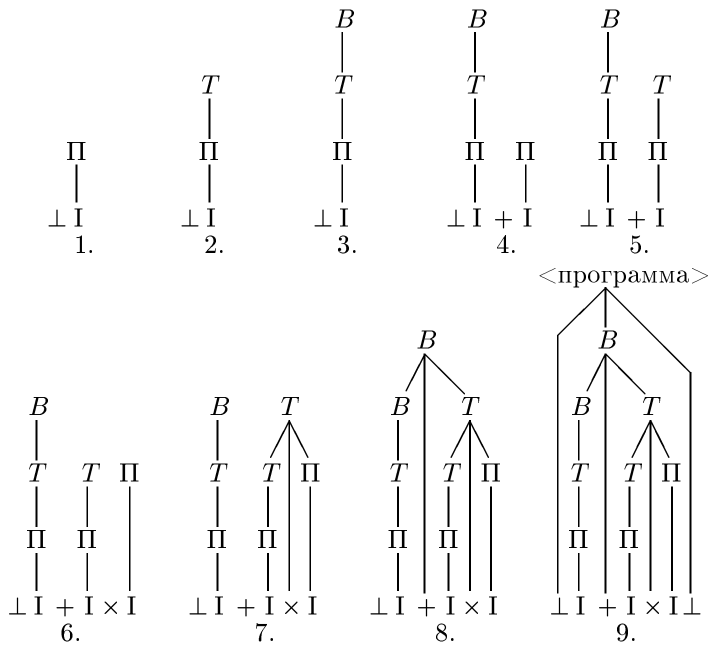
Пример 2.2.1. Рассмотрим для примера грамматику
$G = (V, T, P, \text{<программа>})$
$V = (\text{<программа>}, \perp, B, T, \Pi, I, +, \times, (,)\}$
$T = \{I, +, \times, (,), \perp\}$.
$P = \{$1. $\text{<программа>} \to \perp B \perp$
2. $B \to T$
3. $B \to B + T$
4. $T \to \Pi$
5. $T \to T \times \Pi$
6. $\Pi \to I$
7. $\Pi \to (B)\ \}$

Порождаемые грамматикой G слова можно рассматривать как упрощённые арифметические выражения (соответственно раскрывается смысл нетерминальных символов: B - выражение, T - терм, Π – первичное выражение, I - идентификатор).

Рассмотрим слово $\perp I + I \times I \perp$. Из начального символа <программа> можно вывести лишь $\perp B \perp$. Теперь можно было бы поробовать применить правило $B \to T$, появляется подцель – получить из T символ I, что достижимо последовательностью правил $T \to \Pi, \Pi \to I$, но при этом никак невозможно получить $\perp I+$, даже использовав вместо $T \to \Pi$ правило $T \to T \times \Pi$. Следовательно, нужно использовать другое правило вместо $B \to T$, заменим его правилом $B \to B + T$. Продолжая дальше получим дерево вывода слова $\perp I + I \times I \perp$:



При *анализе снизу-вверх*, наоборот, в цепочке ищут подцепочки, являющиеся правыми частями порождающих правил. Они заменяются соответствующей левой частью правила. Например, для слова $\perp I + I \times I \perp$ из описанной выше грамматики G, получим при левостороннем анализе снизу-вверх следующую последовательность восстановления дерева вывода:



Как правило, описанный анализ снизу-вверх и сверху-вниз, будет иногда делать шаги, которые впоследствии оказываются неверными, а иногда нужно совершать возврат для продолжения анализа. Хотя возможно строить анализаторы снизу-вверх, допускающие возврат, делается это редко. Существуют весьма изящные способы синтаксического анализа снизу-вверх, при которых никаких возвратов не требуется и цепочка, заменяемая левой частью соответствующего правила, определяется однозначно. Естественно, что на грамматику при этом накладываются ограничения, обеспечивающие её эффективное (без возвратов) выявление самой левой основы каждой цепочки. При этом мы получаем более узкие классы КС-грамматик, для которых задача синтаксического анализа эффективно разрешима. Среди таких классов выделяются своей широтой и практической применимостью простые грамматики с отношением предшествования и $LR(k)$-грамматики, описанию свойств которых отведены следующие два раздела.

2.3. Грамматики с отношением предшествования

Отношение предшествования устанавливается между двумя символами грамматики (как терминальными, так и нетерминальными), которые могут встретиться рядом в выводимой цепочке. Существуют следующие три соотношения:

$\lessdot$ (меньше)

$\doteq$ (равно)

$\gtrdot$ (больше)

Определение 2.3.1.

Для заданной КС-грамматики $G = (V, T, P, S)$ отношение предшествования между произвольными символами устанавливается следующим образом:

1. $\alpha \doteq \beta$, если существует порождающее правило $A \to \varphi\alpha\beta\psi$ среди правил P.

2. $\alpha \lessdot \beta$, если существует правило $A \to \varphi\alpha A_1\psi$ и вывод $A_1 \overset{*}{\Rightarrow} \beta\psi_1$ для некоторого $\psi_1 \in V^*$.

3. $\alpha \gtrdot \beta$, если а) существует правило $A \to \varphi A_1\beta\psi$ и вывод $A_1 \overset{*}{\Rightarrow} \varphi_1\alpha$,

либо б) существует правило $A \to \varphi A_1 A_2\psi$ и выводы $A_1 \overset{*}{\Rightarrow} \varphi_1\alpha$, $A_2 \overset{*}{\Rightarrow} \beta\varphi_2$ для некоторых $\psi_1, \psi_2, \varphi_1, \varphi_2 \in V^*$.

Между двумя символами грамматики может существовать от одного до трёх отношений предшествования или не существовать ни одного. Последнее имеет место, когда эти символы не встречаются рядом ни в одной из цепочек, выводимых данной грамматикой. Отношение равенства ($\doteq$) двух символов соответствует возможности их одновременного появления рядом друг с другом. Отношение $\alpha \lessdot \beta$ соответствует появлению сначала α, а позднее β в выводимой цепочке $\varphi\alpha\beta\psi$. Аналогично истолковывается смысл отношения $\gtrdot$.

Пример 2.3.2. Рассмотрим грамматику $G = (V, T, P, S)$,

$T = \{«, », a\}$

$V = T \cup \{S, h\}$

$P : S \to H$ »,

$H \to Ha$

$H \to$ «,

$H \to HS$

Грамматика порождает строки символов a с открывающими « и закрывающими » кавычками. Отношение предшествования удобно задавать в виде матрицы

	S	H	«	»	a
S	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
H	$\doteq$	$\lessdot$	$\lessdot$	$\doteq$	$\doteq$
«	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
»	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
a	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$

В такой матрице справа сверху вниз стоят левые силы отношения, а сверху – правые символы отношения.

Знак отношения между H и « стоит на пересечении строки H и столбца « ($H \lessdot$ «).

Рассмотрим строку «а«а»» и попытаемся восстановить её вывод. Расставим отношения предшествования между символами:

$« \gtrdot a \gtrdot « \gtrdot a \gtrdot » \gtrdot »$

Начнём сворачивание нашей строки слева.

Так как $« \gtrdot a$, следовательно, символ a был получен позже символа « и символ « можно заменить левой частью правила $H \to «$. Получим цепочку $H \doteq a \gtrdot « \gtrdot a \gtrdot » \gtrdot »$. $H \doteq a$, и $a \gtrdot «$, следовательно, сегмент Ha получен в результате применения какого-то правила вывода. Правую часть Ha имеет единственное правило $H \to Ha$, поэтому нашу цепочку можно свернуть к виду

$H \lessdot « \gtrdot a \gtrdot » \gtrdot »$

Теперь сегментом, который можно свернуть, является символ «, который появился «позже» и H и a, стоящих с ним рядом. Продолжая так дальше получим следующую последовательность сворачивания (подчёркиванием выделен сворачиваемый сегмент):

сворачиваемая цепочка	используемое правило
$\underline{«} \gtrdot a\ «a»»$	$H \to «$
$H \doteq a\ «a»»$	$H \to Ha$
$H \lessdot \underline{«} \gtrdot a»»$	$H \to «$
$H \lessdot \underline{H \doteq a} \gtrdot »»$	$H \to Ha$
$H \lessdot \underline{H \doteq »} \gtrdot »$	$S \to H»$
$\underline{H \doteq S} \gtrdot »$	$H \to HS$
$H \doteq »$	$S \to H»$

Используя выписанные справа правила легко восстанавливаем вывод слова $«a«a»»$.

Алгоритм установления отношений предшествования.

Заполнение матрицы отношений предшествования существенно упрощается, если использовать множества $L(U)$ самых левых символов, которые можно получить из символа U, и множество $R(U)$ самых правых символов.

Точнее, множества $L(U)$ и $R(U)$ определяются для произвольных $U \in V$ соотношениями:

$L(U) = \{\alpha | \alpha \in V, \exists\, (\varphi \in V^*)$, такое, что $U \xrightarrow{*} \alpha\varphi\}$
$R(U) = \{\alpha | \alpha \in V, \exists\, (\varphi \in V^*)$, такое, что $U \xrightarrow{*} \varphi\alpha\}$

Для каждого правила $A \to \alpha_1 \alpha_2 \ldots \alpha_m$ грамматики необходимо для каждой пары символов $\alpha_i \alpha_{i+1}$ проставить в матрице предшествования

1) отношение $\alpha_i \doteq \alpha_{i+1}$
2) отношение $\alpha_i \lessdot \beta, \beta \in L(\alpha_{i+1})$
3) отношение $\beta \gtrdot \alpha_{i+1}, \beta \in R(\alpha_i)$
4) отношение $\beta_1 \gtrdot \beta_2, \beta_1 \in R(\alpha_i), \beta_2 \in R(\alpha_{i+1})$

Пример 2.3.3. Для грамматики $G = (V, T, P, S)$

$V = \{S, H, \perp, a\}$
$T = \{\perp, a\}$

P: 1) $S \to H\perp$
2) $H \to \perp$
3) $H \to Ha$
4) $H \to HS$

Множества левых и правых символов следующие:

U	$L(U)$	$R(U)$
S	$\perp, H$	$\perp$
H	$\perp, H$	$a, \perp, S$

Первое правило грамматики даёт отношения:
$H \doteq \perp$ и $\beta \gtrdot \perp$, где $\beta \in R(H)$, т.е.
$a \gtrdot \perp, \perp \gtrdot \perp, S \gtrdot \perp$.
Третье правило приводит к отношениям:
$H \doteq a$, и $\beta \gtrdot a$, где $\beta \in R(H)$, т.е. $a \gtrdot a, \perp \gtrdot a, S \gtrdot a$.

Наконец, четвёртое правило грамматики индуцирует отношения:
$H \doteq S$
$H \lessdot L(S)$, т.е. $H \lessdot H, H \lessdot \perp$
$R(H) \gtrdot S$, т.е. $a \gtrdot S, \perp \gtrdot S, S \gtrdot S$,
$R(H) \gtrdot L(S)$, т.е. $a \gtrdot \perp, \perp \gtrdot \perp, S \gtrdot \perp, a \gtrdot H, \perp \gtrdot H, S \gtrdot H$.

Заполнив таким образом матрицу отношений предшествования получим:

	S	H	a	$\perp$
	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
	$\doteq$	$\lessdot$	$\doteq$	$\lessdot, \doteq$
	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$

Между символами H и $\perp$ два отношения $H \doteq \perp$ и $H \lessdot \perp$, что является отражением двойственной роли символа $\perp$, отмечающего как начало, так и конец строки. Наличие нескольких отношений предшествования для пары символов делает невозможным однозначное выделение сегмента сворачивания.

Определение 2.3.4. Грамматика называется грамматикой с отношением предшествования (ОП-грамматикой), если меду любыми символами существует не более одного отношения предшествования.

Теорема 2.3.5. Для любой КС-грамматики G можно построить эквивалентную ей грамматику с отношением предшествования.

Доказательство. Пусть $G = (V, T, P, S)$ - КС-грамматика. Построим грамматику $G' = (V', T, P', S)$ с тем же множеством терминальных символов T и аксиомой S. Каждое правило грамматики G $A \to \alpha_1 \alpha_2 \dots \alpha_l \in P$ заменим следующей совокупностью правил грамматики G', расширив нужным образом множество V:

а) $A \to \alpha_{P_1} \dots \alpha_{P_l}$,

б) $\alpha_{P_i} \to \alpha_i, i = 1, 2, \dots, l$.

$\alpha_{P_1} \dots \alpha_{P_l}$ - новые нетерминальные символы грамматики G', т.е. каждый символ α_i в правой части правила дублируется новым нетерминальным символом α_{P_i}, причём если даже и встречаются символы $\alpha_i = \alpha_j$, то им ставятся в соответствие разные новые нетерминальные символы α_{P_i}, α_{P_j}.

Совершенно очевидно, что грамматики G и G' эквивалентны. Докажем, что G' - ОП-грамматика.

Рассмотрим любую пару стоящих рядом символов и покажем, что между ними отношение предшествования определяется однозначно. Обозначим эту пару $\alpha\beta$.

1^0. $\alpha \in V, \beta \in V$. Но старые символы можно получить лишь после применения правил типа (б), а это случай 3-б определения отношения предшествования, поэтому $\alpha \gtrdot \beta$.

2^0. $\alpha \in V, \beta_P \in V' - V$, т.е. $\beta = \beta_P$. Это случай либо 3-а, либо 3-б определения отношения предшествования, поэтому $\alpha \gtrdot \beta$.

3^0. $\alpha \in V' - V$, т.е. $\alpha = \alpha_{P_l}, \beta \in V$. Знак $\doteq$, очевидно, вообще не может появиться. Знак $\lessdot$ может появиться лишь так:

$A \xrightarrow{*} \varphi\alpha_{P_i} B\psi$ и $B \xrightarrow{*} \beta z$, поэтому $\alpha_{P_i} \lessdot \beta$, при этом α_{P_i} - не последний символ в последовательности

$\alpha_{P_1} \ldots \alpha_{P_l}$, т.е. $i < l(*)$.

Отношение $\gtrdot$ может появиться по определению лишь следующим образом: $A \to \varphi_1 Bx\varphi_2$ и $B \xrightarrow{*} \psi_1\alpha_{P_i}$, а символ x мог либо совпадать с β, т.е. $x = \beta$, либо из x выводится β справа: $x \xrightarrow{*} \beta\psi_2$.

Из того, что $A \xrightarrow{*} \varphi_1\psi_1\alpha_{P_i}$, получается, что $\alpha_{P_i} = \alpha_{P_l}(**)$, т.е. α_{P_i} - последний символ правила.

Условия (*) и (**) не могут выполняться одновременно, поэтому неоднозначности в отношениях α_{P_i} и β не возникает.

4^0. $\alpha, \beta \in V' = V$, т.е. и α и β новые нетерминальные символы, $\alpha = \alpha_{P_i}$, $\beta = \beta_{q_i}$.

Рассмотрим все возможные случаи отношения предшествования между α и β.

4.1. $\alpha_{P_i} \doteq \beta_{q_i}$. Это возможно лишь при $P = q, j = i = 1$.

4.2. $\alpha_{P_i} \lessdot \beta_{q_i}$. Это возможно лишь при $j = 1, i < l_p$.

4.3. $\alpha_{P_i} \gtrdot \beta_{q_j}$. Это возможно в двух случаях

$$i = l_p, j \geqslant 2, \text{ либо}$$
$$i = l_p, j = 1$$

Условия, при которых определяются разные знаки отношений предшествования не пересекаются. Это указывает на однозначность отношений и в случае 4^0.

Таким образом доказано, что в грамматике G' отношение предшествования определяется однозначно, т.е.

G' - ОП-грамматика.

Вообще говоря, вовсе не обязательно следовать изложенному алгоритму построения грамматике G'.

G		
$S \to H\bot$	$S \to H_1\bot, H_1 \to H_1\bot_1 \to \bot$	$S \to \bar{H}\bot, \bar{H} \to H$
$H \to \bot$	$H \to \bot$	$H \to \bot$
$H \to HS$	$H \to H_2a_2, H_2 \to H, a_2 \to a$	$H \to Ha$
$H \to Ha$	$H \to H_3S_3, H_3 \to H, S_3 \to$	$H \to \hat{H}S, \hat{H} \to H$

Грамматика G' по доказанной теореме является ОП-грамматикой, эквивалентной G. Нетрудно убедиться, что и грамматика G'' эквивалентна G и является ОП-грамматикой. Однако, как в грамматике G', так и в грамматике G'' встречаются правила с одинаковыми правыми частями, что крайне неудобно при синтаксическом анализе – неизвестно, когда какое правило применять. Так при выделении сегмента H в грамматике G'' его можно свернуть к виду $\bar{H}$ и $\hat{H}$, а в грамматике G' неоднозначность ещё больше (H_1, H_2, H_3).

Определение 2.3.6. КС-грамматику с отношением предшествования будем называть простой (КСП), если правые части всех её правил различны, т.е. никакая пара правил не имеет вида $A \to \varphi$, $B \to \varphi$, $A \neq B$.

Опишем теперь более точно алгоритм левостороннего анализа в КСП-грамматике.

Будем называть сегментом цепочку $\alpha_1\alpha_2 \dots \alpha_n$, входящую в цепочку $\varphi\beta_H\alpha_1\alpha_2 \dots \alpha_n\beta_H\psi$, если имеют место отношения $\varphi\beta_H \lessdot \alpha_1 \doteq \alpha_2 \doteq \dots \doteq \alpha_n \gtrdot \beta_k\psi, \alpha_i, \beta_H, \beta_k \in V$. Алгоритм левостороннего анализа в КСП-грамматике состоит в последовательном нахождении самого левого сегмента σ анализируемой цепочки и замене его вхождения (т.е. сворачиванию) левой частью правила грамматики $A \to \sigma$. Если в результате сворачивания мы придём к символу S, то анализируемое слово порождается грамматикой и мы восстановим его правосторонний вывод.

Если на некотором шаге, не достигнув S, алгоритм не сможет продолжать работу, то исходное слово не принадлежит языку.

Теорема 2.3.7. Если в КСП-грамматике дан вывод $A \xrightarrow{*} \varphi$ и ψ - самый левый сегмент φ, то существует правило $B \to \psi$ и вывод $A \xrightarrow{*} \psi' B\psi'' \to \psi'\psi\psi'' = \varphi$ такой, что последний вывод эквивалентен первоначальному.

Доказательство проведём индукцией по длине вывода $A \overset{*}{\to} \varphi$.

1. Пусть длина вывода $n = 1$. Тогда $A \to \varphi$ и это и есть искомое правило.

2. Предположим, что теорема верна для всех $k < n, n > 1$.

3. Докажем теорему для длины вывода n.

Пусть вывод $A \overset{*}{\to} \varphi$ имеет длину n. Разобъём φ на части следующим образом. Пусть на первом шаге вывода $A \overset{*}{\to} \varphi$ получена цепочка $\alpha_1 \ldots \alpha_k$, т.е. $A \to \alpha_1 \ldots \alpha_k$. Затем $\alpha_i \overset{*}{\to} \varphi_i$. Тогда $\varphi = \varphi_1 \varphi_2 \ldots \varphi_k$. Выделим двигаясь от начала цепочки φ те φ_i, для которых $\alpha_i = \varphi_i$, т.е. $\varphi = \alpha_1 \ldots \alpha_m D \varphi_{m+2} \ldots \varphi_k$, где $D = \alpha_{m+1}$ - нетерминальный символ, такой что $D \overset{*}{\to} \beta_1 \ldots \beta_t$. Тогда $\varphi = \alpha_1 \ldots \alpha_m \beta_1 \ldots \beta_t \beta_{t+1} \ldots \beta_{t_k}$.

Покажем, что искомый сегмент ψ находится в цепочке $\beta_1 \ldots \beta_t$. Рассмотрим отношение предшествования между α_m и $\beta_1 : \alpha_m \lessdot \beta_1$ с другой стороны $\beta_t \gtrdot \beta_{t+1}$, следовательно ψ является самым левым сегментом в цепочке $\beta_1 \ldots \beta_t$, т.е. $\beta_1 \ldots \beta_t = \tilde{\varphi}$ и $D \overset{*}{\to} \tilde{\varphi} = \tilde{\psi}' \psi \tilde{\psi}''$, но вывод $D \overset{*}{\to} \tilde{\varphi}$ короче n, следовательно, правило $B \to \psi$ существует по предложению индукции, но это же правило должно быть и для ψ в цепочке φ, следовательно вывод будет:

$$A \to \alpha_1 \alpha_2 \ldots \alpha_m D \alpha_{m+2} \ldots \alpha_k \overset{*}{\to} \alpha_1 \ldots \alpha_m D \varphi_{m+2} \ldots \varphi_k \overset{*}{\to}$$

$$\overset{*}{\to} \alpha_1 \ldots \alpha_m \tilde{\psi}' B \tilde{\psi}'' \varphi_{m+2} \ldots \varphi_k \to$$

$$\to \alpha_1 \ldots \alpha_m \tilde{\psi}' \beta_1 \ldots \beta_t \varphi_{m+2} \ldots \varphi_k = \varphi,$$

что и требовалось доказать.

Таким образом в КСП-грамматике описанный алгоритм анализа, работающий по отношению предшествования, однозначно восстанавливает правосторонний вывод, При практической реализации алгоритма анализа символы анализируемой цепочки записывают в магазин до тех пор, пока между входящим символом a и последним символом магазина α_i имеет место отношение $\lessdot$ или $\doteq$, т.е. $\alpha_i \lessdot a$, или $\alpha_i \doteq a$. Если $\alpha_i \gtrdot a$, то в магазине ищем сегмент.

$\alpha_{j-1} \lessdot \alpha_j \doteq \ldots \doteq \alpha_{i-1} \doteq \alpha_i$.

Цепочка $\alpha_j \dots \alpha_i$ заменяется левой частью единственного правила $A \to \alpha_j \dots \alpha_i$.

Всякую ли грамматику можно преобразовать в эквивалентную КСП-грамматику? Из последней доказанной теоремы легко усмотреть, что грамматика КСП – однозначная (существует единственный правосторонний вывод, который мы и восстанавливали).

Таким образом, ответ на поставленный вопрос – отрицательный, вследствие алгоритмической неразрешимости проблемы распознавания однозначности языка.

2.4. Функции предшествования

В развитых языках программирования число терминальных и нетерминальных символов слишком велико, чтобы можно было хранить матрицу отношений предшествования в памяти ЭВМ в процесе трансляции. Пусть алфавит V состоит из m символов. Оказывается, что вместо m значений отношений предшествования можно ограничиться $2m$ числами – значениями функций предшествования f и g, подобранными так, чтобы отношения между числами – значениями $f(\alpha)$ и $g(\beta)$ давали отношение между α и β.

Точнее, Определение 2.4.1. Функции f и g, определённые на символах множества V со значениями во множестве целых чисел называются функциями предшествования, если из отношения

$\alpha \doteq \beta$ следует $f(\alpha) = g(\beta)$
$\alpha \lessdot \beta \Rightarrow f(\alpha) < g(\beta)$
$\alpha \gtrdot \beta \Rightarrow f(\alpha) > g(\beta)$

Не для всякой ОП-грамматики существует функции предшествования. Это связано с транзитивностью значений функций предшествования и нетранзитивностью отношений предшествования. Пусть, например, отношения предшествования таковы, что $A \lessdot B$, $B \gtrdot B$, $B \doteq C$, $A \gtrdot C$, тогда $f(A) < g(B) < f(B) = g(C) < f(A)$, полученное неравнство противоречиво и не может выполняться. Чтобы убедиться в возможности существования такой цепочки отношений предшествования, достаточно взглянуть на пример.

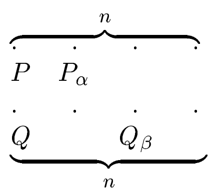
Пример 2.4.2.

$S \to AO«$	$L(\alpha)$	$R(\alpha)$		S	A	O	«	a	»
$S \to «»$	A «	»	S						$\gtrdot$
$A \to «$	«	«	A	$\lessdot$	$\lessdot$	$\doteq$	$\lessdot$	$\lessdot$	
$O \to a$	aSA «	a»	O						$\doteq$
$O \to aS$			«	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\doteq$
$O \to S$			a	$\doteq$	$\lessdot$		$\lessdot$		$\gtrdot$
			»						$\gtrdot$

$f(a) < g() < f() = g() < f(A)$.

Покажем, что существует алгоритм, позволяющий построить функции предшествования, либо выяснить, что их нет.

Возьмём $2n$ точек, если в алфавите V n символов.



Каждой точке из строк P и Q ставится в соответствие элемент из V.

Точки верхнего ряда будем обозначать $P_\alpha, P_\beta, ...$, а точки нижнего ряда $Q_\alpha, Q_\beta, ...\,; \alpha, \beta \in V$.

1. Соединим Q_β с P_α дугой, направленной от Q_β к P_α, если $\alpha \doteq \beta$, либо $\alpha \lessdot \beta$.

$$Q_\beta \longrightarrow P_\alpha$$

2. Соединим P_α с Q_β дугой, направленной от P_α к Q_β, если $\alpha \doteq \beta$, либо $\alpha \gtrdot \beta$.

$$P_\alpha \longrightarrow Q_\beta$$

3. Если между α и β отношения предшествования нет, то вершины P_α и Q_β не связаны.

Подсчитаем, во сколько различных вершин, включая P_α можно попасть, двигаясь из P_α по графу по направлению стрелок от вершины к вершине, обозначим это число $f(\alpha)$.

Подсчитаем, во сколько вершин, считая Q_β, можно попасть, двигаясь аналогично, исходя из Q_β, обозначим это число $g(\beta)$. Докажем, что если в грамматике существуют функции отношения предшествования f^0 и g^0, то определённые выше функции f и g также будут функциями отношений предшествования.

Предположим противное: функции f^0 и g^0 существуют, но $f(\alpha)$ и $g(\beta)$ не согласуются с отношением предшествования. Выясним, где может появиться это несогласование.

1. Если $\alpha \doteq \beta$, то $f(\alpha) = g(\beta)$, так как вершины P_α и Q_β соединены следующим образом:

$$P_\alpha \longleftrightarrow Q_\beta$$

2. Если $\alpha \lessdot \beta$, то $f(\alpha) \leqslant g(\beta)$, так как P_α и Q_β соединены дугой вида

$$P_\alpha \longleftarrow Q_\beta$$

и мы можем попасть из вершины Q_β во все вершины, куда можно попасть, исходя из P_α.

Таким образом, единственный случай несоответствия может получиться, если $\alpha \lessdot \beta$, а $f(\alpha) = g(\beta)$.

Это случай, если P_α и есть путь в Q_β:

$P_{\alpha_1} Q_{\beta_1} P_{\alpha_2} Q_{\beta_2} \ldots Q_{\beta_{n-1}} P_{\alpha_n} Q_{\beta_n}, \alpha_1 = \alpha, \beta_1 = \beta.$

По построению графа имеем отношения:

$\alpha_1 \gtrdot = \beta_1\ \alpha_2 \gtrdot = \beta_2 \ldots \alpha_n \gtrdot = \beta_n$, так как из P_{α_i} пришли в Q_{β_i}

$\alpha_2 \lessdot = \beta_2 \ldots \alpha_n \lessdot = \beta_{n-1}$, т.к. из Q_{β_i} пришли в $P_{\alpha_{i+1}}$

Тогда для функций f^0 и g^0 имеем:

$f^0(\alpha_1) \geqslant g^0(\beta_1) \ldots f^0(\alpha_n) \geqslant g^0(\beta_n)$

$f^0(\alpha_2) \leqslant g^0(\beta_1) \ldots f^0(\alpha_n) \leqslant g^0(\beta_{n-1})$

или переписывая все неравенства в строку:

$f^0(\alpha_1) \geqslant g^0(\beta_1) \geqslant \ldots \geqslant f^0(\alpha_n) \geqslant g^0(\beta_n)$, т.е.

$f^0(\alpha_1) \geqslant g^0(\beta_n)$ и так как $\alpha_1 = \alpha$, $\beta_n = \beta$,
то $f^0(\alpha) \geqslant g^0(\beta)$, что противоречит условию $\alpha \lessdot \beta$.

3. Случай $\alpha \gtrdot \beta$ рассматривается совершенно аналогично случаю 2.

Таким образом, мы доказали, что функции $f(\alpha)$ и $g(\beta)$ либо согласуются с отношением предшествования и являются функциями предшествования, либо функций предшествования не существует.

Для практического нахождения функций предшествования удобно пользоваться следующим алгоритмом:

1^0. $f(\alpha) := g(\alpha) := 1$ для всех $\alpha \in V$.

2^0. Для всех $\alpha, \beta \in V$ проделать следующие операции:

если $\alpha \gtrdot \beta$ и $f(\alpha) \leqslant g(\beta)$, то $\quad f(\alpha) := g(\beta) + 1$;

если $\alpha \lessdot \beta$ и $f(\alpha) \geqslant g(\beta)$, то $\quad g(\beta) := g(\alpha) + 1$;

$\alpha \doteq \beta \quad \min\{f(\alpha, g(\beta)\} := \max\{f(\alpha, g(\beta)\}$

3^0. Если значение какой-либо функции, полученное на очередном этапе согласования в пункте 2^0 больше $2n$, то функций предшествования не существует.

Указанный алгоритм аналогичен описанной процедуре поиска числа проходимых вершин и потому достигает нужного результата, либо сигнализирует об отсутствии функций предшествования (число пройденных вершин не может быть больше $2n$).

При синтаксическом анализе с помощью отношения предшествования мы одновременно контролировали выводимость цепочки: если встречались два символа, стоящие рядом, между которыми нет отношения предшествования, то из S такое слово заведомо получить нельзя. Заменив отношения предшествования функциями предшествования, мы тем самым потеряли возможность синтаксического контроля, поскольку функции предшествования определены для любой пары символов, и ошибка выяснится лишь при попытке свернуть цепочку.

Замечание 2.4.3. Метод отношения предшествования применим и к грамматикам общего типа, правила которых имеют вид:

$\varphi \to \psi, \varphi, \psi \in V^*, \varphi \neq \varnothing$.

Множества $L(\alpha)$ самых левых и $R(\alpha)$ самых правых символов, выводимых из символов α определяются совершенно аналогично случаю КС-грамматик:

$$L(\alpha) = \{\beta : \alpha\varphi \to \beta\psi \in P, \text{ либо } \\ \alpha\varphi \to \beta_1\psi \in P \text{ и } \beta \in L(\beta_1)\}$$

$$R(\alpha) = \{\beta : \varphi\alpha \to \psi\beta \in P, \text{ либо } \\ \varphi\alpha \to \psi\beta_1 \in P \text{ и } \beta \in R(\beta_1)\},$$

здесь φ и ψ - цепочки, возможно пустые, в алфавите V. $\alpha, \beta, \beta_1 \in V$.

Отношение предшествования определяется следующим образом:

$\alpha \doteq \beta$, если $\varphi \to \psi_1\alpha\beta\psi_2 \in P$

$\alpha \lessdot \beta$, если $\varphi \to \psi_1\alpha\beta_1\psi_2 \in P$ и $\beta \in L(\beta_1)$

$\alpha \gtrdot \beta$, если

1. $\varphi \to \psi_1\alpha_1\beta\psi_2 \in P \quad \alpha \in R(\alpha_1)$
2. $\varphi \to \psi_1\alpha_1\beta_1\psi_2 \in P \quad \alpha \in R(\alpha_1), \beta \in L(\beta_1)$.

Пример 2.4.4.
$G = (V, T, P, S)$
$T = \{a, b\}$
$V = T \cup \{S, D, H, B, A\}$

$$\text{Множество } P \text{ правил: } \begin{array}{ll} S \to ASD & B \to b \\ S \to Aba & bD \to bBa \\ aD \to Ha & A \to a \\ H \to D & \end{array}$$

Порождаемый язык есть $\{a^n b^n a^n | n \geqslant 1\}$.

Матрица отношений предшествования для грамматики G имеет вид:

$L(\alpha)$	$F(\alpha)$		S	B	A	H	D	a	b
A, H, D, a	D, a	S					$\doteq$		
b	b	B				$\lessdot$	$\lessdot$	$\doteq$	
a, H, D	a	A	$\doteq$	$\doteq$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$
D	D, a	H				$\lessdot$	$\lessdot$	$\doteq$	
	a	D				$\gtrdot$	$\gtrdot$	$\gtrdot$	
H, D		a	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
b		b		$\doteq$		$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$

Синтаксический анализ производится в точности как и ранее.

2.5. Детерминированные автоматы с магазинной памятью

Для генерации (порождения или допущения) правильных предложений (слов) языка имеются две конструкции: грамматики и автоматы. Грамматику можно использовать для анализа, если существует отношение предшествования. Пусть язык порождается (допускается) автоматом. Как в этом случае эффективно решается задача синтаксического анализа? Рассмотренные ранее автоматы с магазинной памятью были недетерминированными и могли зайти в тупик, что ещё не означает недопустимости слова автоматом.

Рассмотрим теперь некоторый специальный тип магазинных автоматов, называемых детерминированными. Говоря неформально, речь идёт о таких автоматах, в которых для каждой ситуации однозначно определяется следующая. Эти автоматы представляют практический интерес, поскольку обычно они отвергают или допускают цепочку быстрее, чем недетерминированные автоматы. Языки, допускаемые такими автоматами, легче поддаются грамматическому анализу.

Определение 2.5.1. Магазинный автомат ДНА $= \{Z, K, T, \delta, z_0, q_0, \&\}$, где & - закон окончания работы, соответствующий окончанию работы $F-$ или $N-$ автомата, называется *детерминированным*, если для всех $q \in K$ выполняются следующие условия:

либо 1) $\delta(z, q, \varnothing) = (z', r)$
однозначная функция, т.е. множество $(\delta, q, \varnothing)$
состоит из одного элемента, и
$(\delta, q, \varnothing) = \varnothing$ для всех $a \in T$.

либо 2) $\delta(z, q, a) = (z', r)$ - однозначная функция
для всех $a \in T$
и $(z, q, \varnothing) = \varnothing$,

$z' \in Z^*, r \in K.$

Определение 2.5.1 утверждает, что автомат, находясь в определённом состоянии q и обозревая определённый символ z в магазинной памяти, либо всегда читает входной символ – и при этом, прочтя определённый символ, может перейти лишь в одну ситуацию, зависящую от этого символа (условие 2),- либо вообще может перейти лишь в одну ситуацию и не может читать никакого входного символа. В магазинной памяти детерминированного автомата всегда должна быть записана некоторая цепочка, чтобы был возможен следующий такт работы автомата.

Лемма 2.5.2. Для каждого детерминированного автомата ΔHA можно построить эквивалентный ему детерминированный автомат $\Delta HA'$, такой, что символ $\hat{z}_0$ магазина всегда находится в начале магазина до окончания работы автомата.

Доказательство. Действительно, каждому правилу

$\delta(z_0, q_0, a) = (z_1 \dots z_k, r),$

либо

$\delta(z_0, q_0, \varnothing) = (z_1 \dots z_k, r),$

автомата ДНА поставим в соответствие правила

$\delta'(z_0, q_0, a) = (\hat{z}_0 z_1 \dots z_k, r),$

либо

$\delta'(z_0, q_0, \varnothing) = (\hat{z}_0 z_1 \dots z_k, r).$

Дополним определение функции δ' правилами $\delta'(\hat{z}_0, q_0, \varnothing) = (\varnothing, q)$ для всех $q \in K$. Для остальных ситуаций функция δ' совпадает с функцией δ. Если ДНА F-автомат, то для допустимого им слова x имеем работу:

$$(z_0, q_0, x) \vdash (z', q, x') \overset{*}{\vdash} (z'', q_F, \varnothing).$$

Соответствующая работа ДНА'-автомата будет:

$$(z_0, q_0, x) \vdash (\hat{z}_0 z', q, x') \overset{*}{\vdash} (\hat{z}_0 z'', q_F, \varnothing).$$

следовательно, слово x допустимо и ДНА$'$-автоматом. Аналогично доказывается и обратное утверждение, т.е. F-автоматы ДНА и ДНА$'$ эквивалентны.

Если ДНА N-автомат, то для допустимого им слова имеем работу

$$(z_0, q_0, x) \vdash (z', q, x') \overset{*}{\vdash} (\varnothing, r, \varnothing).$$

однозначно определяемая соответствующая работа автомата ДНА$'$

$$(z_0, q_0, x) \vdash (\hat{z}_0 z', q, x') \overset{*}{\vdash} (\hat{z}_0 \ldots \hat{z}_0, r, \varnothing) \overset{*}{\vdash} (\varnothing, r, \varnothing),$$

следовательно, слово x допускается и ДНА$'$-автоматом. Очевидно и обратное. Таким образом произвольные автоматы ДНА и ДНА$'$ эквивалентны.

В силу доказанной эквивалентности будем в дальнейшем рассматривать лишь детерминированные автоматы, у которых до окончания работы в магазине записана непустая цепочка.

Теорема 2.5.3. Любой детерминированный автомат эквивалентен детерминированному автомату, совершающему только элементарные действия, а именно

1. $\delta(z, q, a) = (z, r)$ - читает один символ и переходит в новое состояние.
2. $\delta(z, q, \varnothing) = (\varnothing, r)$ - стирает один символ.
3. $\delta(z, q, \varnothing) = (zz', r), z' \in Z$ - записывает один символ в магазин.

Доказательство. Построим по заданному детерминированному автомату ДНА $= \{Z, K, T, \delta, z_0, q_0, \&\}$ элементарный детерминированный автомат (ДЭА-автомат).

ДЭА $= \{Z, K', T, B', z_0, q_0, \&\}$.

Для этого каждое правило функционирования ДНА заменим строго определённой последовательностью правил ДЭ-автомата.

I. Пусть в состоянии q, читая символ x в магазине ДНА работал согласно правилу:

$\delta(z, q, \varnothing) = (z', r), z' = z_1 \ldots z_k$.

Тогда правила ДЭ-автомата будут:

$\delta'(z, q, \varnothing) = (z, q_{z_1 \ldots z_k r})$,

$\delta'(z, q_{z_1 \ldots z_k r}, \varnothing) = (\varnothing, q_{z_1 \ldots z_k r})$

1. $\delta'(z_{-1}, q_{z_1 \ldots z_k r}, \varnothing) = (z_{-1} z_1, q_{z_2 \ldots z_k r}), \forall\, z_{-1} \in Z$
2. $\delta'(z_1, q_{z_2 \ldots z_k r}, \varnothing) = (z_1 z_2, q_{z_3 \ldots z_k r})$

$\vdots$

$k.\quad \delta(z_{k-1}, q_{z_k r}, \varnothing) = (z_{k-1} z_k, q_r)$

$k+1$. $\delta(z_k, q_r, \varnothing) = (z_k, r)$

$q_{z_i z_{i+1} \dots z_k r} \notin K, i = 1, 2, \dots, k$ - новые введённые состояния автомата ДЭА.

Таким образом, работе

$(z''z, q, \varnothing) \vdash (z''z_1 \dots z_k, r, \varnothing)$

автомата ДНА соответствует единственно возможная последовательность работ ДЭ-автомата:

$(z''z, q, \varnothing) \vdash (z''z, q_{z_1 \dots z_k r}, \varnothing) \vdash$

$\vdash (z'', q_{z_1 \dots z_k r}, \varnothing) \vdash (z''z_1, q_{z_2 \dots z_k r}, \varnothing) \overset{*}{\vdash}$

$\overset{*}{\vdash} (z''z_1 \dots z_k, q_r, \varnothing) \vdash (z''z_1 \dots z_k, r, \varnothing)$

и, наоборот, любой работе

$(z, q, \varnothing) \overset{*}{\vdash} (z'', r, \varnothing)(q, r \in K)$

(т.е. не являются новыми введёнными состояниями автомата ДЭА) соответствует в точности одна последовательность ситуаций автомата ДНА.

II. Пусть в состоянии q, читая символ z в магазине, ДНА работал согласно правилу:

$\delta(z, q, a) = (z', r), z' = z_1 \dots z_k$, т.е. -

ДНА всегда читал входной символ.

Тогда правила ДЭА-автомата будут:

$\delta'(z, q, a) = (z, q_{z_1 \dots z_k r})$.

Остальные правила в точности повторяют правила, записанные в пункте I: правила $1, 2, \dots, k, k+1$.

Эквивалентность автоматов ДНА и ДЭА вполне очевидна.

Вообще говоря детерминированные F и N автоматы не эквивалентны, так как детерминированный N-автомат не допускает язык, содержащий наряду со словом $x \neq \varnothing$ слово $xy(y \neq \varnothing)$, т.е. допустимое детерминированным N-автоматом слово x не может быть начальным словом другого допустимого слова.

Замечание 2.5.4. От подобного ограничения легко избавиться введением специального символа конца слова ω, не принадлежащего терминальным символам языка, добавив к правилам грамматики правило $S_0 \to S\omega$, где $S, \omega \notin V$. Так как ω не может стоять нигде внутри слова, то никакое слово уже не может быть начальным словом другого слова.

Детерминированные F-автоматы свободны от такого недостатка. Автоматы, допускающие языки L_ω с символом конца слова, обозначим F_ω и N_ω.

Теорема DNF 2.5.5. Для всякого детерминированного N-автомата существует эквивалентный ему F-автомат. Действительно, в этом случае, аналогично лемме 2.5.2., легко ввести символ $\tilde{z}_0$, стоящий перед z_0 в магазине (т.е. магазин содержит символы - $\tilde{z}_0 z_0$ ..., а символ z_0 всегда находится в магазине по соглашению 2.5.2). Когда N-автомат окончит работу, в магазине останется символ $\tilde{z}_0$ и тогда применяется дополнительно введённое правило

$\tilde{z}_0, p, \varnothing) \vdash (\varnothing, q_F, \varnothing), p \in K, q_F \in F.$

Так построенный автомат будет детерминированным F-автоматом, эквивалентным исходному N-автомату.

Теорема о существовании детерминированного N-автомата эквивалентного детерминированному F-автомату неверно по указанным выше соображениям, однако на множестве L_ω КС-языков эти автоматы эквивалентны.

Теорема $DF_\omega N_\omega$ 2.5.6. Для всякого детерминированного F_ω-автомата существует эквивалентный ему N_ω-автомат.

Сохранив все правила F_ω-автомата $F_\omega = \{Z, K, T, \delta, F, z_0, q_0\}$ добавим переход в состояние стирания магазина ($\bar{q}$) после чтения символа ω.

$\delta_1 : (z, q, \omega) \vdash (z, \bar{q}, \varnothing)$

и правила, обеспечивающие стирание содержимого магазина

$\delta_2 : (z, \bar{q}, \varnothing) \vdash (\varnothing, \bar{q}, \varnothing)$

Построенный N_ω-автомат

$N_\omega = \{Z, K, T, \delta', z_0, q_0\}$

будет детерминированным и эквивалентным F_omega-автомату.

Обладают ли языки, допускаемые детерминированными автоматами какими-нибудь дополнительными свойствами, упрощающими их анализ? Оказывается, что языки, допускаемые ДНА – однозначные. Сейчас по заданному ДНА будет построена грамматика, порождающая язык L(ДНА) и доказана её однозначность.

Теорема 2.5.7. Для любого детерминированного N-автомата существует однозначная КС-грамматика G, такая, что $L(N) = L(G)$.

Доказательство: В NG-теореме было уже доказано существование КС-грамматики G, такой, что $L(N) = L(G)$. Нам фактически нужно сейчас доказать лишь однозначность построенной там КС-грамматики для случая детерминированного N-автомата.

Итак, пусть $N = \{Z, K, T, \delta, F, z_0, q_0\}$ - детерминированный N-автомат. Построим КС-грамматику $G = (V, T, P, S)$, $V = T \cup \{A_{zpq}\} \cup \{S\}$, где A_{zpq} - нетерминальный символ (z - символ магазинной памяти автомата, $p, q \in K$). Смысл этого нетерминального символа, как и раньше, заключается в том, что $A_{zpq} \xrightarrow{*} x$, если $(z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$, т.е. автомат в состоянии p, имея в магазине символ z, прочитывает слово x и обнуляет магазин, то слово x должно выводиться из нетерминального символа A_{zpq}. Для этого множество P правил грамматики определим следующим образом:

P_1. $S \to A_{zpq}$ для всех $p \in K$.

P_2. $A_{zpq} \to a$, если $(z, p, a) \vdash (\varnothing, q, \varnothing), a \in T$.

P_3. $A_{zpq} \to aA_{z_k r_k r_{k-1}} \ldots A_{z_2 r_2 r_1} A_{z_1 r_1 q}$, если

$(z, p, a) \vdash (z_1 \ldots z_k, r_k, \varnothing)$ и существует $y_k \in T$, такой что

$(Z_k, r_k, y_k) \overset{*}{\vdash} (\varnothing, r_{k-1}, \varnothing)$

$y_{k-1} \in T \qquad (Z_{k-1}, r_{k-1}, y_{k-1}) \overset{*}{\vdash} (\varnothing, r_{k-2}, \varnothing)$

$\ldots \qquad \ldots$

$y_1 \in T \qquad (Z_1, r_1, y_1) \overset{*}{\vdash} (\varnothing, q, \varnothing)$

В частности

$A_{zpq} \to \varnothing$, если $(z, p, \varnothing) \vdash (\varnothing, q, \varnothing)$.

1. Пусть $(Z_0, r_0, y_0) \overset{*}{\vdash} (\varnothing, q, \varnothing), x \in T^*$. Докажем, что тогда $S \xrightarrow{*} x$.

Имеем $S \to A_{z_0 p_0 q_0}$.

Докажем, что если $(Z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$, то $A_{zpq} \overset{*}{\to} x$ и указанный вывод – единственно возможный. Приведём для этого индукцию по числу n тактов работы автомата.

1. $n = 1$, т.е. $(z, p, a) \vdash (\varnothing, q, \varnothing), a \in T, z \in Z$.

Тогда по пункту 2 построения множества P существует правило $A_{zpq} \to a$. Рассматриваемый автомат – детерминированный, следовательно переход из ситуации (z, p, a) в ситуацию $(\varnothing, q, \varnothing)$ один и только один, поэтому и соответствующий ему вывод в грамматике – однозначно определён.

2. Пусть утверждение справедливо для любой работы автомата, содержащей менее n тактов.

3. Докажем справедливость утверждения для работы длины n.

Если $n > 1$, то это означает, что первый такт работы имеет вид:

$(z, p, ax) \vdash (z_1 \dots z_k, r_k, x')$, где

$ax' = x \in T^*$ - слово на входе автомата.

Тогда слово $x' = y_k y_{k-1} \dots y_1$ состоит из таких символов y_i, для которых $(z_i, r_i, y_i) \overset{*}{\vdash} (\varnothing, r_{i-1}, \varnothing)$.

Следовательно, имеем цепочку (единственную, в силу детерминированности автомата)

$(z, p, ax') \vdash (z_1 \dots z_k, r_k, y_k \dots y_1) \overset{*}{\vdash} (z_1 \dots z_{k-1}, r_{k-1}, y_{k-1} y_{k-2} \dots y_1) \overset{*}{\vdash} \dots \overset{*}{\vdash} (z_1, r_1, y_1) \overset{*}{\vdash} (\varnothing, q, \varnothing)$, и переход $(z_i, r_i, y_i) \overset{*}{\vdash} (\varnothing, r_{i-1}, \varnothing)$ занимает менее $n - 1$ тактов, поэтому существует единственный вывод

$A_{a_i r_i r_{i-1}} \overset{*}{\to} y_i$

соответственно, единственный вывод

$A_{zpq} \to a A_{z_k r_k r_{k-1}} \dots A_{z_1 r_1 q}$

есть

$A_{zpq} \overset{*}{\to} a y_1 y_{k-1} \dots y_2 y_1 = x$.

2. Пусть $S \overset{*}{\to} x$. Тогда $(z_0, p(?)_0, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$.

Первый шаг вывода непременно имеет вид

$S \to A_{z_0 p_0 q_0}$

Надо доказать, что если $A_{z_0 p_0 q_0} \overset{*}{\to} x$, то существует единственная работа автомата такая, что

$(z_0, q_0, z) \overset{*}{\vdash} (\varnothing, p, \varnothing)$.

Снова будем применять индукцию по длине n вывода.

1. $n = 1$, т.е. $A_{zpq} \to x$, но тогда $x = a \in T$ и $(z, p, a) \vdash (\varnothing, q, \varnothing)$ единственно возможная работа автомата.

2. Предположим, что утверждение справедливо для всех выводов длины, меньше n.

3. Докажем справедливость утверждения для вывода длины n.

Пусть $x = ax'$.

Тогда первый шаг вывода имеет вид

$A_{zpq} \to aA_{z_k r_k r_{k-1}} A_{z_{k-1} r_{k-1} r_{k-2}} \dots A_{z_1 r_1 q}$

При этом в автомате существуют символы $y_k \dots y_1$ такие, что y_k выводится из $A_{z_k r_k r_{k-1}}$, $A_{z_k r_k r_{k-1}} \overset{*}{\to} y_k$.

Но указанные выводы короче n, поэтому существует переход

$(z_k, r_k, y_k) \overset{*}{\vdash} (\varnothing, r_{k-1}, \varnothing)$.

А тогда

$(z, p, x) \vdash (z_1 \dots z_k, r_k, x') \overset{*}{\vdash} (z_1 \dots z_{k-1}, r_{k-1}, y_{k-1} \dots y_1) \overset{*}{\vdash}$
$\overset{*}{\vdash} (z_1, r_1, y_1) \overset{*}{\vdash} (\varnothing, q, \varnothing)$.

Вследствие детерминированности автомата выписанная последовательность переходов – единственная.

3. Итак, доказано, что каждой работе автомата соответствует единственный вывод и наоборот. Установленное взаимно-однозначное соответствие работ автоматаи выводов в грамматике непосредственно приводит к однозначности построенной грамматики, так как автомат детерминированный (что неоднократно использовалось в пунктах 1 и 2 доказательства).

Замечание 2.5.3. Особенно простой получается однозначная грамматика, соответствующая элементарному детерминированному N-автомату. Действительно, в этом случае существуют три типа правил функционирования автомата.

1. $(z, p, a) \vdash (z, q, \varnothing)$
2. $(z, p, \varnothing) \vdash (\varnothing, q, \varnothing)$

3. $(z, p, \varnothing) \vdash (zz', q, \varnothing), z, z' \in Z$.

и правила вывода имеют вид:

α. $S \to A_{z_0 p_0 q}$ для всех $p \in K$.

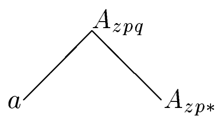
β. $A_{zpq} \to aA_{zpq}$, если $(z, p, a) \vdash (z, q, \varnothing) \overset{*}{\vdash} (\varnothing, r, \varnothing)$
γ. $A_{zpq} \to \varnothing$, если $(z, p, \varnothing) \vdash (\varnothing, q, \varnothing)$
δ. $A_{zpq} \to A_{z'qs}A_{zsr}$, если

$(z, p, \varnothing) \vdash (zz', q, \varnothing) \overset{*}{\vdash} (z, s, \varnothing) \overset{*}{\vdash} (\varnothing, r, \varnothing)$.

2.5.9. Интересно более подробно выяснить порядок восстановления дерева вывода по работе элементарного автомата, чтобы убедиться в его единственности. Дерево вывода восстанавливается не сразу. Действительно, в момент применения правила перехода

$(z, p, a) \vdash (z, q, \varnothing)$

то есть в момент появления вершин A_{zpq}, мы не можем определить неизвестные нам нетерминальные символы (отмеченные звёздочкой)



Будем фиксировать этапы работы автомата по правилам типов 1, 2, 3. 1, 3, 1,$\underline{2}$, 1, 1, 2 – соответствующая работа со словом x на входе цепочка переходов автомата. До тех пор, пока не встретилось правило типа 2, мы ничего не можем сказать о вершинах дерева. После применения правила 2 до ближайшего слева правила 3 дерево восстанавливается. Последнее правило 3, стоящее перед 2, превратится в правило 1, либо 2, следующее правило 2 восстанавливает дерево до ближайшего правила 3 и так далее.

В ОП-грамматике, двигаясь по слову, мы расставляли отношения символами и выделяли сегменты сворачивания. Аналогичная конструкция присутствует и в рассматриваемом случае.

2.5.10. Если внимательно проследить доказательство теоремы 2.5.7, то нетрудно заметить, что однозначность грамматики вытекает не столько из детерминированности автомата, сколько из единственности цепочки переходов при допущении каждого слова. Можно доказать, что язык тогда и только тогда однозначен, когда существует допускающий его автомат такой, что существует единственно возможная цепочка переходов при допущении каждого слова.

Пример. $L = \{xx^R\}, x \in \{a,b\}^*$. Построим N-автомат, допускающий язык $L_1 = L - \varnothing$. Автомат построим недетерминированный с двумя состояниями: в состоянии q_w автомат записывает в магазин читаемый им символ; в любой момент он может либо сохранить состояние q_w, либо перейти в состояние q_{er}, в котором символ магазина сравнивается с входным символом и стирается, если символы совпадают. Точнее.

$N = \{\{a,b\}, \{q_w, q_{er}\}, \{a,b\}, \delta, z_0, q_w\}$

$$
\begin{array}{llll}
\delta: & \delta 1.\ (z_0, q_w, \alpha) & \vdash (\alpha, q_w, \varnothing) & \alpha \in T \\
& \delta 2.\ (\alpha, q_w, \beta) & \vdash (\alpha\beta, q_w, \varnothing) & \alpha, \beta \in T \\
& \delta 3.\ (\alpha, q_w, \alpha) & \vdash (\varnothing, q_{er}, \varnothing) & \alpha \in T \\
& \delta 4.\ (\alpha, q_{er}, \alpha) & \vdash (\varnothing, q_{er}, \varnothing) & \alpha \in T
\end{array}
$$

Следуя NG-теореме построим грамматику G.

$G = (V, T, P, S)$
$V = T \cup \{S\} \cup \{A_{zpq}\}$

P:	$S \to A_{z_0 q_w q_w}$ - непродуктивное правило	
	$S \to A_{z_0 q_w q_{er}}$	
	$A_{z_0 q_w q_{er}} \to \alpha A_{\alpha q_w q_{er}}$	- из правила δ 1
	$A_{\alpha q_w q_{er}} \to \beta A_{\beta q_w q_{er}} A_{\alpha q_{er} q_{er}}$	- из правила δ 2
	$A_{\alpha q_w q_{er}} \to \alpha$	- из правила δ 3
	$A_{\alpha q_{er} q_{er}} \to \alpha$	- из правила δ 4

Устраняя непродуктивные правила и переобозначая нетерминальнгые символы, получим:

$S \to aA$
$S \to bB$
$A \to bBA'$
$A \to aAA'$
$B \to aAB'$
$B \to bBB'$
$A \to a$

$B \to b$
$A' \to a$
$B' \to b$

Легко увидеть, что полученная грамматика однозначна – следствие «однозначной» работы автомата по допущению каждого слова.

2.6. LR(k)-грамматики

LR(k)-грамматики – это наиболее общий вид грамматики для которых существует эффективный левосторонний анализ снизу-вверх, конструкцию которого можно получить механически из грамматики.

Грамматика является LR(k)-грамматикой в том случае, если правило сворачивания подцепочки однозначно определяется всей слева от неё расположенной частью цепочки и k справа расположенными терминальными символами.

В ОП-грамматике мы заглядываем вперёд на один символ, чтобы проставить отношение предшествования и выделить сворачиваемый сегмент. ДЛЯ ДНА этого не нужно. Допускаемый ДНА язык – это LR(0) язык. Язык, порождаемый КСП-грамматикой – LR(1)-язык.

Что же можно получить, заглядывая на k символов вперёд?

Определение 2.6.1. Пусть в КС-грамматике имеется правосторонний вывод некоторой цепочки (не обязательно в терминальных символах) и пусть в этом выводе

$$S \xrightarrow{*} \alpha_1 \ldots \alpha_j A_p a_1 \ldots a_k b_1 \ldots b_m \to \underline{\alpha_1 \ldots \alpha_n}\ \underline{\underline{a_1 \ldots a_k}}$$

на последнем шаге применялась подстановка

$A_p \to \alpha_{j+1} \ldots \alpha_n$.

Пусть дан вывод некоторой другой цепочки (тоже правосторонний), имеющий вид

$$S \xrightarrow{*} \alpha_1 \ldots \alpha_j A_p a_1 \ldots a_k c_1 \ldots c_m \to \underline{\alpha_1 \ldots \alpha_n}\ \underline{a_1 \ldots a_k}$$

и части $\alpha_1 \ldots \alpha_n$ совпадают, и совпадают следующие за ними k символов. Тогда на последнем шаге применялась та же подстановка, что и в первом случае. Такая грамматика называется LR(k)-грамматикой, порождаемый ею язык - LR(k)-языком.

Выполняя анализ слова с использованием магазина, левосторонний анализатор должен просматривать весь магазин (не только фиксированное число символов в нём) и следующие k символов слова, при этом правосторонний вывод восстанавливается однозначно. Фактически язык, эффективно анализируемый любым ранее рассмотренным способом, является LR(k)-языком для некоторого k. Таким образом, соответствие грамматики условиям LR(k) является так же наиболее общим тестом однозначности, который сейчас имеется.

Пример 2.5.2.

$$
\begin{array}{lll}
1.\ P: & S \to Ac & T = \{a, c, d\} \\
 & S \to Bd & V_N = \{A, B, S\} \\
 & & V = V_N \cup T \\
 & A \to Aa & G_1 = (V, T, P, S) \\
 & B \to Ba & \\
 & A \to a & L(G_1) = \{a^n d\} \cup \{a^n c\} n \geqslant 1 \\
 & B \to a. &
\end{array}
$$

G_1 не является LR(k)-грамматикой ни для какого $k = 1, 2, ...$ Действительно, чтобы восстановить вывод слова x, нужно обязательно знать, какой символ стоит в конце. С другой стороны, грамматика G_2 с правилами

$S \to aA$
$A \to aA$
$A \to c$
$A \to d$

порождающая тот же язык, является LR(0)-грамматикой. Поэтому можно сказать, что $\{a^n d\} \cup \{a^n c\}$ - LR(0)-язык.

2. Грамматика G_3 с правилами

1. $S \to SS$
2. $S \to Ac$
3. $S \to Bd$
4. $A \to aA_1$

5. $A_1 \to aa$
6. $B \to aB_1$
7. $B_1 \to aa$

порождает язык $L = \{(a^3\alpha)^{2n}\}$, где α есть либо c, либо d. G_3 является LR(4)-грамматикой, возможность заглянуть на 4 символа вперёд однозначно определяет по значению $\alpha = c$ применение правила 2, а по значению $\alpha = d$ - правила 3.

3. Простая грамматика с отношением предшествования – LR(1)-грамматика. Заглядывая на один символ вперёд, и сравнивая его по предшествованию с символом магазина, однозначно определяем сегмент и правило сворачивания.

4. У LR(k)-грамматики ярко проявляется свойство «несимметричности»:

язык $\{a^n b^n \omega\} \cup \{a^n b^{2n} c\omega\}$ не порождается никакой LR(k)-грамматикой и LR(0)-грамматикой;

язык $\{\omega b^n a^n\} \cup \{\omega c b^{2n} a^n\}$ - LR(0)-язык.

Ниже будет доказано, что для всякой LR(k)-грамматики можно построить детерминированный автомат, допускающий тот же язык (в предположении, что в языке каждое слово кончается символом ω – конца слова). Это может показаться на первый взгляд удивительным, поскольку в общем случае существует бесконечное множество цепочек $\alpha_1\alpha_2 \dots \alpha_n$, которые могут появиться в магазине; таким образом, существует бесконечное число цепочек, которые необходимо использовать для вынесения решения при анализе, в то время как автомат использует только последний элемент магазина, состояние и входной символ.

Всё же благодаря тому, что число порождающий правил конечно, на каждом шаге анализа существует только конечное число возможных действий, независимо от символов $\alpha_1\alpha_2 \dots \alpha_{n-1}$ магазина. Поэтому необходимо только классифицировать возможные цепочки в классы, называемые «состояниями» γ_p, и показать, что если грамматика является LR(k)-грамматикой, можно задать однозначное отображение

$(\alpha_i, \gamma_p, a_j) \vdash (\alpha', \gamma_a)$

Здесь мы, конечно, упростили процесс, чтобы лучше разобраться в идее; символы α_i в магазине должны быть сами сложными множествами, чтобы сделать возможным определение однозначного отображения.

Теорема 2.6.3. Для всякой LR(k)-грамматики G можно построить детерминированный автомат, допускающий язык $L(G)$ (в предположении 2.5.4).

Доказательство.

1. Будем называть состоянием тройку (p, j, x), где

$p : A_p \to \alpha_{p_1} \dots \alpha_{p_{n_p}}$ - правило,

x - слово длины k (если действительной длины не хватает, то дополним символами ω).

$1 \leqslant j \leqslant n_p$ - индекс символа в правой части правила p.

Каждому p можно поставить в соответствие несколько троек.

Пусть $\gamma = \{(p, j, x)\}$ - некоторое множество состояний. Построим множество γ_ψ для слова ψ следующим образом. Пусть слово ψ возникает при некотором правостороннем выводе

$$S \xrightarrow{*} \alpha_1 \dots \alpha_i A_p x x' \to \alpha_1 \dots \alpha_i \alpha_{p_1} \dots \alpha_{p_n} x x' = \\ = \psi \alpha_{p_{j+1}} \dots \alpha_{p_n} x x'$$

ψ - это правая часть слова, включающая j символов от цепочки, выведенной из A_p. Тогда состояние (p, j, x) включается в множество γ_ψ.

Хотя количество слов ψ бесконечно, количество различных множеств γ_ψ конечно, так же существует лишь конечное число правил с конечным числом букв. Будем теперь строить множество для слова $\psi\alpha_{j+1}$.

Если α_{j+1} не входит в $\alpha_{p_1} \dots \alpha_{p_n}$, то p не входит в $\gamma_{\psi\alpha_{j+1}}$.

Если же α_{j+1} входит в $\alpha_{p_1} \dots \alpha_{p_n}$, то в $\gamma_{\psi\alpha_{j+1}}$ входит тройка $(p, j+1, x)$.

Далее процесс продолжается аналогично.

Возможна ситуация, когда α_{j+1} - это символ d_1 из цепочки $d_1 \dots d_t$, полученной из нетерминального символа α_{j+2} при подстановке q.

В таком случае в $\gamma_{\psi\alpha}$ включается $(q, 0, y)$.

y - первые k символов из цепочки $\alpha_{j+3} \dots \alpha_{n_p} x$. Это означает, что нужно свернуть по правилу q следующие ? символов, но прежде проверить, нет ли там другого состояния $(r, 0, z)$. Каждой паре γ и p поставим в соответствие множество слов x, таких, что $(p, n_p, x) \in \gamma$ и обозначим это множество H_{γ_p}

$H_{\gamma_p} = \{x : (p, n_p, x) \in \gamma\}$.

Очевидно, что $H_{\gamma_p} \cap H_{\gamma_q} = \varnothing$, если $p \neq q$.

2. Строим автомат.

Пусть дано слово $a_1 \dots a_n = w$.

Сначала мы находимся в нулевой позиции отмеченной точкой для каждого правила $.a_1 \dots a_n$, затем движемся по слову $a_1 \dots a_i . a_{i+1} \dots a_n$, в конце w - пустое слово.

Автомат D.

Начальное состояние $(p_S, 0, \varnothing)$.

$Z = \{\gamma\} \cup V$

$K = \{(x)\} \cup \{r_{px}\}, (x)$ - слово длины k в терминальных символах.

$T = T$

q_0 - начальное состояние,

$z_0 = \gamma_0$

В начале работы (т.е. преобразования ситуации (γ_0, q_0, w) в ситуацию $(\gamma_0, (x), w')$ автомата

$(\gamma_0, q_0, w) \overset{*}{\vdash} (\gamma_0, (x), w')$

считывается k символов из слова w. Затем начинается его «нормальное» функционирование. Рассмотрим среднее звено работы:

$\dots \overset{*}{\vdash} (\gamma_0 \alpha_1 \gamma_1 \alpha_2 \gamma_2 \dots \alpha_n \gamma_n, (x), w'') \overset{*}{\vdash} \dots$

Автомат работает следующим образом.

$\gamma_1 = \gamma_{\alpha_1}$ - это множество γ_ψ от $\psi = \alpha_1, \gamma_2 = \gamma_{\alpha_1 \alpha_2}$ от $\psi = \alpha_1, \alpha_2, \dots, \gamma_n = \gamma_\psi$ от $\psi = \alpha_1 \dots \alpha_n$.

Автомат совершает преобразования:

а) если $x \notin H_{\gamma_n p}$, то в магазин записывается символ из x (левый) и $\gamma_{\psi x}$ и в x справа добавляется символ из w'', а первый символ x исчезает (он уже перешёл в магазин).

б) если же $x \in H_{\gamma_n p}$, то это значит, что надо сворачивать, а это делается с помощью служебного состояния r_{px}.

$(\gamma_0\alpha_1\gamma_1 \ldots \alpha_n\gamma_n, (x), w'') \vdash$

$(\gamma_0\alpha_1\gamma_1 \ldots \alpha_n\gamma_n, r_{px}, w'') \vdash \ldots$

автомат стирает нужные α, γ и затем пишет:

$\ldots \vdash (\gamma_0\alpha_1\gamma_1 \ldots \alpha_i\gamma_i A_p\gamma_p, (x), w'')$

и после этого всё делается снова.

В силу единственности правостороннего вывода в LR(k)-грамматике и детерминированности автомата эквивалентность языков $L(G)$ и $L(D)$ можно считать доказанной.

Заинтересованный читатель, руководствуясь описанной конструкцией детерминированного автомата, легко построит элементарный детерминированный автомат, допускающий все слова простой грамматики с отношением предшествования, где при выводе нужно заглядывать вперёд на один символ.

Для доказательства того, что язык допускаемый детерминированным автоматом есть LR(0)-язык, нам понадобится следующая лемма.

Пусть G_N - грамматика, построенная в теореме 2.5.7. (грамматика эквивалентная детерминированному автомату).

Лемма 2.6.4. Пусть $A_{zpq} \xrightarrow{*} x$ и некоторое подслово слова ???. Тогда эквивалентны следующие утверждения:

а) в дереве вывода $A_{zpq} \xrightarrow{*} x$ содержится поддерево вывода $A_{z'p'q'} \xrightarrow{*} x'$;

б) при работе автомата $(z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$ осуществляется подработа $(z', p', x') \overset{*}{\vdash} (\varnothing, q', \varnothing)$.

Доказательство ведётся индукцией по длине n вывода $A_{zpq} \xrightarrow{*} x$, равной, как отмечалось выше, числу тактов при работе автомата $(z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$.

Ясно, что детального анализа требуют лишь случаи, когда (при выводе в) из а)) поддерево вывода $A_{z'p'q'} \xrightarrow{*} x'$ не совпадает с деревом вывода $A_{zpq} \xrightarrow{*} x$ и (при выводе а) из б)) подработа $(z', p', x') \overset{*}{\vdash} (\varnothing, q', \varnothing)$ не совпадает со всей работой $(z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$.

При $n = 1$ дерево вывода $A_{zpq} \to x$ не имеет отличных от него самого поддеревьев с нетерминальной вершиной, а работа автомата $(z, p, x) \vdash (\varnothing, q, \varnothing)$ не индуцирует отличных от неё самой подработ вида $(z', p', x') \vdash (\varnothing, q', \varnothing)$.

Пусть утверждение леммы справедливо для всех выводов $A_{zpq} \overset{*}{\to} x$, длина которых меньше n, докажем его для n.

Пусть верно а) Вывод $A_{zpq} \overset{*}{\to} x$ имеет вид

$$(*) \quad A_{zpq} \to aA_{z_k q_k q_{k-1}} A_{z_{k-1} q_{k-2} q_{k-2}} A_{z q_1 q} \overset{*}{\Rightarrow}$$
$$\overset{*}{\Rightarrow} x = ax_k x_{k-1} \dots x_1,$$

где каждое x_i получается при этом выводе из соответствующего $A_{z_i q_i q_{i-1}}$ (считаем, что $q = q_0$). Поддерево вывода $A_{z'p'q'} \overset{*}{\to} x'$ является поддеревом одного из деревьев $A_{z_i q_i q_{i-1}} \overset{*}{\to} x_i$, однако длина вывода $A_{z_i q_i q_{i-1}} \overset{*}{\to} x_i$ меньше n, следовательно, работа автомата $(z_i, p_i, x_i) \overset{*}{\vdash} (\varnothing, q_{i-1}, \varnothing)$ индуцирует подработу $(z', p', x') \overset{*}{\vdash} (\varnothing, q', \varnothing)$. В силу (*) и детерминированности автомата первый такт его работы в конфигурации (z, p, x) имеет вид

$$(z, p, x) \vdash (z_1 z_2 \dots z_k, q_k, x_k x_{k-1} \dots x_2 x_1),$$

а дальнейшая работа должна протекать по следующей схеме:

$$(z_1 z_2 \dots z_{k-1} z_k, q_k, x_k x_{k-1} \dots x_2 x_1) \overset{*}{\vdash}$$
$$(z_1 z_2 \dots z_{k-1}, q_{k-1}, x_{k-1} \dots x_2 x_1) \overset{*}{\vdash} \dots \overset{*}{\vdash}$$
$$(z_1 z_2, q_2, x_2 x_1) \overset{*}{\vdash} (z_1, q_1, x_1) \overset{*}{\vdash} (\varnothing, q, \varnothing)$$

Следовательно, работа $(z, p, x) \overset{*}{\vdash} (\varnothing, q, \varnothing)$ индуцирует подработу

$$(z_1 \dots z_i, q_i, x_i \dots x_1) \overset{*}{\vdash} (z_1 \dots z_{i-1}, q_{i-1}, x_{i-1} \dots x_1),$$

эквивалентную ей подработу $(z_i, p_i, x_i) \vdash (\varnothing, q_{i-1}, \varnothing)$ и, следовательно, и подработу $(z', p', x') \vdash (\varnothing, q', \varnothing)$.

Пусть верно в). Первый шаг работы автомата имеет вид $(z, p, x) \vdash (z_1 \dots z_k, r_3, y)$, где $x = ay$, а схема последующей его работы такова (зафиксированы моменты, когда в магазине стираются символы $z_k, z_{k-1}, \dots, z_1$):

$(z_1 \dots z_{k-1} z_k, q_k, y_k) \overset{*}{\vdash} (z_1 \dots z_{k-1}, q_{k-1}, y_{k-1}) \overset{*}{\vdash} \dots \overset{*}{\vdash}$

$(z_1, q_1, y_1) \overset{*}{\vdash} (\varnothing, q_0, \varnothing)$

Здесь считается $r = q_k$, $q = q_0$, $y = y_k y_{k-1} \dots y_1$. Так как подработа $(z', p', x') \overset{*}{\vdash} (\varnothing, q', \varnothing)$ стирает в магазине только символ z', то она является подработой одной из работ

$(z_1 \dots z_i, q_i, y_i) \overset{*}{\vdash} (z_1 \dots z_{i-1}, q_{i-1}, y_{i-1})$.

Теорема 2.6.5. Грамматика G_N, построенная в 2.5.7., есть LR(0)-грамматика.

Доказательство. Пусть

$S \overset{*}{\to} \alpha_1 \alpha_2 \dots \alpha_i A b_1 b_2 \dots b_k \to \alpha_1 \alpha_2 \dots \alpha_m b_1 b_2 \dots b_k = \varphi$

$S \overset{*}{\to} \alpha_1 \alpha_2 \dots \alpha_m c_1 c_2 \dots c_l = \psi$

два правосторонних вывода, а

$\alpha_1 \alpha_2 \dots \alpha_m \overset{*}{\to} u$

какой-либо вывод слова, состоящего лишь из терминальных символов, из слова $\alpha_1 \alpha_2 \dots \alpha_m$. Этим выводом порождается вывод и некоторого слова x' из слова $\alpha_{i+1} \dots \alpha_m$, а, следовательно, и из символа A:

$A \to \alpha_{i+1} \dots \alpha_m \overset{*}{\to} x'$

Обозначим $u b_1 \dots b_k = x_1$, $u c_1 c_2 \dots c_l = x_2$.

A есть некоторое $A_{z'p'q'}$. Теперь дважды применяем лемму 2.6.4. Для некоторого q_1 дерево вывода

$(S \to) A_{z_0 p_0 q_0} \overset{*}{\to} x_1$

содержит поддерево $A_{z'p'q'} \to x'$. Следовательно, работа автомата $(z_0, p_0, x_1) \overset{*}{\vdash} (\varnothing, q_1, \varnothing)$ индуцирует подработу $(z', p', x') \overset{*}{\vdash} (\varnothing, q', \varnothing)$. В силу детерминированности автомата и совпадения начала u слов x_1 и x_2, содержащего подслово x', та же подработа $(z', p', x') \overset{*}{\vdash} (\varnothing, q', \varnothing)$ осуществится и при чтении автоматом слова x_2, т.е. при работе $(z_0, p_0, x_2) \overset{*}{\vdash} (\varnothing, q_2, \varnothing)$. Следовательно, в дереве вывода

$(S \to) A_{z_0 p_0 q_2} \overset{*}{\to} \alpha_1 \dots \alpha_i \alpha_{i+1} \dots \alpha_m c_1 \dots c_l \overset{*}{\to} v x' c_1 \dots c_l = x_2$

содержится поддерево вывода

$(A =)A_{z'p'q'} \to \alpha_{i+1} \ldots \alpha_m \xrightarrow{*} x'$

В силу того, что мы ограничились правосторонним выводом слова ψ, последний раз в этом выводе должно применяться правило

$A \to \alpha_{i+1} \ldots \alpha_m$

из чего следует, что грамматика G_N есть LR(0)-грамматика.

Таким образом, показано (теорема 2.6.3), что для всякой LR(k)-грамматики, порождающей слова с маркером конца слова, можно построить детерминированный автомат, допускающий тот же язык. Например, КСП-грамматика есть LR(1)-грамматика, используя теорему 2.6.3 можно построить эффективный анализатор для КСП в виде детерминированного автомата. С другой стороны, теорема 2.5.5. утверждает, что всякий детерминированный N-автомат допускает LR(0) язык. Таким образом, доказано, что LR(k)грамматики, $k \geqslant 1$, эквивалентны LR(0) грамматикам для языков с отметкой конца слова.