

# **Лекция 3-4. Компьютерная лингвистика**

## Основные понятия и определения

**Определение:** *алфавит* - это конечное множество символов.

Предполагается, что термин "символ" имеет достаточно ясный интуитивный смысл и не нуждается в дальнейшем уточнении.

**Определение:** *цепочкой символов в алфавите  $V$*  называется любая конечная последовательность символов этого алфавита.

**Определение:** цепочка, которая не содержит ни одного символа, называется *пустой цепочкой*. Для ее обозначения будем использовать символ  $\epsilon$ .

Более формально цепочка символов в алфавите  $V$  определяется следующим образом:

- (1)  $\epsilon$  - цепочка в алфавите  $V$ ;
- (2) если  $\alpha$  - цепочка в алфавите  $V$  и  $a$  - символ этого алфавита, то  $\alpha a$  - цепочка в алфавите  $V$ ;
- (3)  $\beta$  - цепочка в алфавите  $V$  тогда и только тогда, когда она является таковой в силу (1) и (2).

**Определение:** если  $\alpha$  и  $\beta$  - цепочки, то цепочка  $\alpha\beta$  называется *конкатенацией* (или *сцеплением*) цепочек  $\alpha$  и  $\beta$ .

Например, если  $\alpha = ab$  и  $\beta = cd$ , то  $\alpha\beta = abcd$ .

Для любой цепочки  $\alpha$  всегда  $\alpha\varepsilon = \varepsilon\alpha = \alpha$ .

**Определение:** *обращением* (или *реверсом*) цепочки  $\alpha$  называется цепочка, символы которой записаны в обратном порядке.

Обращение цепочки  $\alpha$  будем обозначать  $\alpha^R$ .

Например, если  $\alpha = abcdef$ , то  $\alpha^R = fedcba$ .

Для пустой цепочки:  $\varepsilon = \varepsilon^R$ .

**Определение:**  $n$ -ой степенью цепочки  $\alpha$  (будем обозначать  $\alpha^n$ ) называется конкатенация  $n$  цепочек  $\alpha$ .

$$\alpha^0 = \epsilon; \alpha^n = \alpha\alpha^{n-1} = \alpha^{n-1}\alpha.$$

**Определение:** *длина цепочки* - это число составляющих ее символов.

Например, если  $\alpha = abcdefg$ , то длина  $\alpha$  равна 7.

Длину цепочки  $\alpha$  будем обозначать  $|\alpha|$ . Длина  $\epsilon$  равна 0.

**Определение:** *язык* в алфавите  $V$  - это подмножество цепочек конечной длины в этом алфавите.

**Определение:** обозначим через  $V^*$  множество, содержащее все цепочки конечной длины в алфавите  $V$ , включая пустую цепочку  $\epsilon$ .

Например, если  $V = \{0, 1\}$ , то  $V^* = \{\epsilon, 0, 1, 00, 11, 01, 10, 000, 001, 011, \dots\}$ .

**Определение:** обозначим через  $V^+$  множество, содержащее все цепочки конечной длины в алфавите  $V$ , исключая пустую цепочку  $\epsilon$ .

Следовательно,  $V^* = V^+ \cup \{\epsilon\}$ .

Ясно, что каждый язык в алфавите  $V$  является подмножеством множества  $V^*$ .

**Определение:** *декартовым произведением*  $A \times B$  множеств  $A$  и  $B$  называется множество  $\{ (a,b) \mid a \in A, b \in B \}$ .

**Определение:** *порождающая грамматика*  $G$  - это четверка  $(VT, VN, P, S)$ , где

$VT$  - алфавит *терминальных символов (терминалов)*,

$VN$  - алфавит *нетерминальных символов (нетерминалов)*, не пересекающийся с  $VT$ ,

$P$  - конечное подмножество множества  $(VT \cup VN)^+ \times (VT \cup VN)^*$ ; элемент  $(\alpha, \beta)$  множества  $P$  называется *правилом вывода* и записывается в виде  $\alpha \rightarrow \beta$ ,

$S$  - *начальный символ (цель)* грамматики,  $S \in VN$ .

Для записи правил вывода с одинаковыми левыми частями

$$\alpha \rightarrow \beta_1 \quad \alpha \rightarrow \beta_2 \quad \dots \quad \alpha \rightarrow \beta_n$$

будем пользоваться сокращенной записью

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n.$$

Каждое  $\beta_i$ ,  $i = 1, 2, \dots, n$ , будем называть *альтернативой* правила вывода из цепочки  $\alpha$ .

Пример грамматики:  $G1 = (\{0,1\}, \{A,S\}, P, S)$ , где  $P$  состоит из правил

$$S \rightarrow 0A1$$

$$0A \rightarrow 00A1$$

$$A \rightarrow \epsilon$$

**Определение:** цепочка  $\beta \in (VT \cup VN)^*$  непосредственно выводима из цепочки  $\alpha \in (VT \cup VN)^+$  в грамматике  $G = (VT, VN, P, S)$  (обозначим  $\alpha \rightarrow \beta$ ), если  $\alpha = \xi_1 \gamma \xi_2$ ,  $\beta = \xi_1 \delta \xi_2$ , где  $\xi_1, \xi_2, \delta \in (VT \cup VN)^*$ ,  $\gamma \in (VT \cup VN)^+$  и правило вывода  $\gamma \rightarrow \delta$  содержится в  $P$ .

Например, цепочка 00A11 непосредственно выводима из 0A1 в грамматике G1.

**Определение:** цепочка  $\beta \in (VT \cup VN)^*$  выводима из цепочки  $\alpha \in (VT \cup VN)^+$  в грамматике  $G = (VT, VN, P, S)$  (обозначим  $\alpha \Rightarrow \beta$ ), если существуют цепочки  $\gamma_0, \gamma_1, \dots, \gamma_n$  ( $n \geq 0$ ), такие, что  $\alpha = \gamma_0 \rightarrow \gamma_1 \rightarrow \dots \rightarrow \gamma_n = \beta$ .

**Определение:** последовательность  $\gamma_0, \gamma_1, \dots, \gamma_n$  называется выводом длины  $n$ .

Например,  $S \Rightarrow 000A111$  в грамматике G1 (см. пример выше), т.к. существует вывод  $S \rightarrow 0A1 \rightarrow 00A11 \rightarrow 000A111$ . Длина вывода равна 3.

**Определение:** языком, порождаемым грамматикой  $G = (VT, VN, P, S)$ , называется множество  $L(G) = \{\alpha \in VT^* \mid S \Rightarrow \alpha\}$ .

Другими словами,  $L(G)$  - это все цепочки в алфавите VT, которые выводимы из S с помощью P.

Например,  $L(G1) = \{0^n 1^n \mid n > 0\}$ .

**Определение:** цепочка  $\alpha \in (VT \cup VN)^*$ , для которой  $S \Rightarrow \alpha$ , называется *сентенциальной формой* в грамматике  $G = (VT, VN, P, S)$ .

Таким образом, язык, порождаемый грамматикой, можно определить как множество терминальных сентенциальных форм.

**Определение:** грамматики  $G_1$  и  $G_2$  называются *эквивалентными*, если  $L(G_1) = L(G_2)$ .

Например,

$G_1 = (\{0,1\}, \{A,S\}, P_1, S)$

и

$G_2 = (\{0,1\}, \{S\}, P_2, S)$

$P_1: S \rightarrow 0A1$

$P_2: S \rightarrow 0S1 \mid 01$

$0A \rightarrow 00A1$

$A \rightarrow \varepsilon$

эквивалентны, т.к. обе порождают язык  $L(G_1) = L(G_2) = \{0^n 1^n \mid n > 0\}$ .

**Определение:** грамматики  $G_1$  и  $G_2$  почти эквивалентны, если  $L(G_1) \cup \{\epsilon\} = L(G_2) \cup \{\epsilon\}$ .

Другими словами, грамматики почти эквивалентны, если языки, ими порождаемые, отличаются не более, чем на  $\epsilon$ .

Например,

$G_1 = (\{0,1\}, \{A,S\}, P_1, S)$

и

$G_2 = (\{0,1\}, \{S\}, P_2, S)$

$P_1: S \rightarrow 0A1$

$P_2: S \rightarrow 0S1 \mid \epsilon$

$0A \rightarrow 00A1$

$A \rightarrow \epsilon$

почти эквивалентны, т.к.  $L(G_1) = \{0^n 1^n \mid n > 0\}$ , а  $L(G_2) = \{0^n 1^n \mid n \geq 0\}$ , т.е.  $L(G_2)$  состоит из всех цепочек языка  $L(G_1)$  и пустой цепочки, которая в  $L(G_1)$  не входит.



## Классификация грамматик и языков по Хомскому

(грамматики классифицируются по виду их правил вывода)

### ТИП 0:

Грамматика  $G = (VT, VN, P, S)$  называется *грамматикой типа 0*, если на правила вывода не накладывается никаких ограничений (кроме тех, которые указаны в определении грамматики).

### ТИП 1:

Грамматика  $G = (VT, VN, P, S)$  называется *неукорачивающей грамматикой*, если каждое правило из  $P$  имеет вид  $\alpha \rightarrow \beta$ , где  $\alpha \in (VT \cup VN)^+$ ,  $\beta \in (VT \cup VN)^+$  и  $|\alpha| \leq |\beta|$ .

Грамматика  $G = (VT, VN, P, S)$  называется *контекстно-зависимой (КЗ)*, если каждое правило из  $P$  имеет вид  $\alpha \rightarrow \beta$ , где  $\alpha = \xi_1 A \xi_2$ ;  $\beta = \xi_1 \gamma \xi_2$ ;  $A \in VN$ ;  $\gamma \in (VT \cup VN)^+$ ;  $\xi_1, \xi_2 \in (VT \cup VN)^*$ .

*Грамматику типа 1* можно определить как неукорачивающую либо как контекстно-зависимую.

Выбор определения не влияет на множество языков, порождаемых грамматиками этого класса, поскольку доказано, что множество языков, порождаемых неукорачивающими грамматиками, совпадает с множеством языков, порождаемых КЗ-грамматиками.

## ТИП 2:

Грамматика  $G = (VT, VN, P, S)$  называется *контекстно-свободной (КС)*, если каждое правило из  $P$  имеет вид  $A \rightarrow \beta$ , где  $A \in VN$ ,  $\beta \in (VT \cup VN)^+$ .

Грамматика  $G = (VT, VN, P, S)$  называется *укорачивающей контекстно-свободной (УКС)*, если каждое правило из  $P$  имеет вид  $A \rightarrow \beta$ , где  $A \in VN$ ,  $\beta \in (VT \cup VN)^*$ .

Грамматику типа 2 можно определить как контекстно-свободную либо как укорачивающую контекстно-свободную.

Возможность выбора обусловлена тем, что для каждой УКС-грамматики существует почти эквивалентная КС-грамматика.

## ТИП 3:

Грамматика  $G = (VT, VN, P, S)$  называется *праволинейной*, если каждое правило из  $P$  имеет вид  $A \rightarrow tB$  либо  $A \rightarrow t$ , где  $A \in VN$ ,  $B \in VN$ ,  $t \in VT$ .

Грамматика  $G = (VT, VN, P, S)$  называется *леволинейной*, если каждое правило из  $P$  имеет вид  $A \rightarrow Bt$  либо  $A \rightarrow t$ , где  $A \in VN$ ,  $B \in VN$ ,  $t \in VT$ .

Грамматику типа 3 (регулярную, *P-грамматику*) можно определить как праволинейную либо как леволинейную.

Выбор определения не влияет на множество языков, порождаемых грамматиками этого класса, поскольку доказано, что множество языков, порождаемых праволинейными грамматиками, совпадает с множеством языков, порождаемых леволинейными грамматиками.

## **Соотношения между типами грамматик:**

- (1) любая регулярная грамматика является КС-грамматикой;
- (2) любая регулярная грамматика является УКС-грамматикой;
- (3) любая КС- грамматика является УКС-грамматикой;
- (4) любая КС-грамматика является КЗ-грамматикой;
- (5) любая КС-грамматика является неукорачивающей грамматикой;
- (6) любая КЗ-грамматика является грамматикой типа 0.
- (7) любая неукорачивающая грамматика является грамматикой типа 0.
- (8) любая УКС-грамматика является грамматикой типа 0.

Формально задача восстановления грамматики состоит главным образом в построении процедур восстановления синтаксических правил неизвестной грамматики по конечному множеству предложений или цепочек  $S_t$  языка  $L(G)$ , порождаемого грамматикой  $G$ , и, возможно, по конечному множеству цепочек из дополнения к  $L(G)$ . Восстановленная грамматика — это совокупность правил, описывающая данное конечное множество цепочек из  $L(G)$ , по которой можно предсказать другие цепочки той же природы (в том или ином смысле), что и цепочки заданного множества. На рис. 7.1 показана основная блок-схема устройства восстановления грамматики. Мету качества восстановленной грамматики определяют таким образом, чтобы она давала в некотором смысле удовлетворительные результаты. В последние годы было

предложено несколько мер качества, определенных в терминах сложности восстановленной грамматики, которые нашли применение в задачах восстановления грамматики.

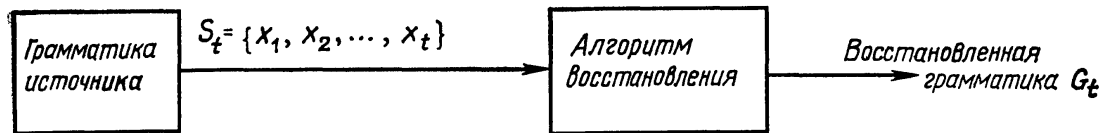


Рис. 7.1. Основная модель устройства восстановления грамматики.

**Определение 7.1.** Последовательность цепочек множества  $\{+y \mid y \in L\} \cup \{-y \mid y \in V_T^* - L\}$  называется информационной последовательностью  $I(L)$  языка  $L$ . Информационную последовательность, состоящую только из цепочек языка  $L$ , назовем положительной информационной последовательностью  $I^+(L)$  языка  $L$ . Аналогично, информационную последовательность, состоящую только из цепочек множества  $V_T^* - L$ , назовем отрицательной информационной последовательностью  $I^-(L)$  языка  $L$ .

**Определение 7.2.** Информационная последовательность языка  $L$  называется полной, если

- 1)  $I^+(L)$  содержит все цепочки языка  $L$ , и
- 2)  $I^-(L)$  содержит все цепочки, не принадлежащие  $L$ .

Будем говорить, что положительная информационная последовательность  $I^+(L)$  полна, если каждая цепочка языка  $L$  содержится в  $I^+(L)$ .

**Определение 7.3.** Образцом  $S_t(L)$  языка  $L$  называется множество  $\{+y_1, \dots, +y_t\} \cup \{-y_1, \dots, -y_t\}$ , причем множество  $\{+y_1, \dots, +y_t\}$  называется положительным образцом, а множество  $\{-y_1, \dots, -y_t\}$  — отрицательным образцом. Положительный образец  $S_t$  языка  $L(G)$  называется структурно полным, если для любого правила подстановки из грамматики  $G$  существует хотя бы одна цепочка из  $S_t$ , при построении которой было использовано это правило.

**Определение 7.7.** Грамматика  $G_t$  называется совместимой с образцом языка  $S_t$ , если  $L(G_t)$  содержит все цепочки из положительной части и не содержит ни одной цепочки из отрицательной части образца.



## 15. Восстановление формальных грамматик

Ранее были рассмотрены вопросы обучения на примере задач распознавания образов. При этом объекты описывались векторами вещественных признаков. То есть исходное представление информации имело численную природу. Помимо этого широко распространена проблема обучения в рамках символьных представлений. Здесь существует большое разнообразие методов в зависимости от выходного представления, в качестве которого могут выступать фреймы, системы продукций, формальные грамматики, деревья решений и т.д. Мы рассмотрим проблему обучения в рамках формальных грамматик.

Напомним, что формальная грамматика – это четверка  $G = (V_T, V_N, P, S)$ , где  $V_T$  – алфавит терминальных символов;  $V_N$  – алфавит нетерминальных символов, причем  $V_N \cap V_T = \emptyset$ ;  $V_N \cup V_T = V$  – алфавит грамматики  $G$ ;  $P$  – множество правил подстановки;  $S$  – начальный символ,  $S \in V_N$ .

Как и в случае численных данных, обучение формальным грамматикам основывается по некоторой исходной информации. Здесь обучающей выборкой является совокупность символьных строк, которая может содержать как положительные, так и отрицательные примеры.

Пусть  $\Gamma_0$  – некий изучаемый язык. Введем следующие определения.

*Информационной последовательностью*  $I(\Gamma_0)$  языка  $\Gamma_0$  будем называть последовательность цепочек, каждая из которых принадлежит одному из множеств  $\{\alpha^+ | \alpha^+ \in \Gamma_0\}$  или  $\{\alpha^- | \alpha^- \in V_T^* \setminus \Gamma_0\}$  с указанием того, к какому множеству принадлежит та или иная цепочка. Последовательность цепочек языка  $\{\alpha^+ | \alpha^+ \in \Gamma_0\}$  будем называть *положительной информационной последовательностью*  $I^+(\Gamma_0)$ , а последовательность цепочек из дополнения языка  $\{\alpha^- | \alpha^- \in V_T^* \setminus \Gamma_0\}$  будем называть *отрицательной информационной последовательностью*  $I^-(\Gamma_0)$ .

Информационная последовательность  $I(\Gamma_0)$  называется *полной*, если  $I^+(\Gamma_0)$  содержит все цепочки языка  $\Gamma_0$ , а  $I^-(\Gamma_0)$  содержит все цепочки, не принадлежащие языку  $\Gamma_0$ .

Грамматика  $G$  согласована с грамматикой  $G_0$ , если порождаемые ими языки совпадают  $\Gamma(G) = \Gamma(G_0)$ .

Пусть  $C = \{G_i\}$  – класс грамматик. Класс языков  $\Gamma(C) = \{\Gamma(G) \mid G \in C\}$  называется *идентифицируемым*, если для любой грамматики  $G \in C$  и любой полной информационной последовательности  $I(\Gamma(G))$  существует некоторое число  $N$  и алгоритм, который бы, получая на входе подпоследовательность  $I(\Gamma(G))$ , содержащую не менее  $N$  цепочек, на выходе давал бы грамматику, согласованную с грамматикой  $G$ .

Наряду с понятием информационной последовательности используется понятие образца (или выборки) языка  $\Gamma_0$ . *Образцом*  $S_t$  языка  $\Gamma_0$  будем называть последовательность цепочек  $\{\alpha_i\}_{i=1}^t$ , для каждой цепочки которой известно, принадлежит ли она языку  $\Gamma_0$  или его дополнению  $V_T^* \setminus \Gamma_0$ . *Положительным образцом* будем называть множество  $S_t^+ = S_t \cap \Gamma_0$ , а *отрицательным образцом* – множество  $S_t^- = S_t \cap (V_T^* \setminus \Gamma_0)$ .

Грамматика  $G$  называется *совместимой* с образцом  $S_I$ , если она порождает все положительные примеры этого образца и не порождает ни одного отрицательного примера.

*Структурно полный* образец  $S_I(\Gamma(G))$  языка  $\Gamma(G)$  – это образец, содержащий такие цепочки, при построении которых каждое правило подстановки грамматики  $G$  использовалось хотя бы по одному разу.

Структурная полнота образца является необходимым условием возможности восстановления всех правил грамматики, в то время как полнота информационной последовательности может выступать в качестве достаточного условия идентифицируемости некоторых классов языков. На практике структурная полнота образца достигается существенно легче, чем полнота информационной последовательности, но обоснование структурной полноты также возможно далеко не во всех случаях (чтобы убедиться в этом, попробуйте составить структурно полный образец русского языка).

При решении задачи грамматического вывода различают *текстуальное* (текстовое) представление, при котором имеются лишь положительные примеры, и *информаторное* представление, при котором есть как положительные, так и отрицательные примеры.

Если дан образец языка или информационная последовательность, то проблема обучения, очевидно, заключается в построении грамматики, совместимой с данным образцом, или в идентификации некоторой истинной грамматики. Иными словами, требуется построить формальную грамматику, которая бы как можно лучше описывала структуру языка  $\Gamma_0$ .

Существует два варианта постановки задачи восстановления грамматик.

В первой формулировке (на которую будем ссылаться как на *проблему согласования*) предполагается, что есть некоторая истинная грамматика  $G_0$ , и требуется по информационной последовательности построить такую грамматику  $G$ , которая была бы согласована с грамматикой  $G_0$ , то есть  $\Gamma(G) = \Gamma(G_0)$ .

Во второй формулировке (на которую будем ссылаться как на *проблему грамматического вывода*) считается, что по образцу  $S_t$  необходимо построить такую грамматику  $G$ , которая бы порождала все цепочки положительного образца  $S_t^+$  (и, возможно, бесконечное множество других цепочек) и не порождала цепочки отрицательного образца  $S_t^-$  (и, возможно, бесконечное множество других цепочек), то есть была бы совместима с этим образцом.

Обычно выделяют два класса алгоритмов восстановления грамматик: перечислением и индукцией. Эти два класса алгоритмов имеют определенную связь с двумя приведенными формулировками задачи восстановления грамматик. Сначала мы кратко рассмотрим восстановление грамматик перечислением.

### Восстановление грамматик перечислением

При формулировании проблемы согласования грамматик предполагается, что существует некая истинная грамматика, и целью является нахождение грамматики, согласованной с этой истинной грамматикой. Требование согласованности является очень сильным, так что вызывает сомнение возможность достижения этой цели в достаточно общем случае. Возникает вопрос, при каких ограничениях проблема согласования разрешима?

При теоретическом исследовании этого вопроса оказывается необходимым предполагать, что даны *полные* информационные последовательности (либо только положительная, либо положительная и отрицательная).

Требование полноты позволяет строить теоремы о существовании алгоритмов восстановления грамматик при различных условиях и давать их формальные доказательства. К сожалению, полных информационных последовательностей на практике не встречается. Доказательства существования алгоритмов обычно неконструктивны (либо в них строятся алгоритмы, непригодные из-за проблемы комбинаторного взрыва). Такие теоремы полезны тем, что говорят, для решения каких вариантов задачи грамматического вывода алгоритмы, в принципе, не стоит искать.

### Эвристические процедуры грамматического вывода

Рассмотрим задачу грамматического вывода. В этой задаче не обязательно строить грамматику, согласованную с некоторой истинной грамматикой, а достаточно вывести грамматику, совместимую с данным образцом языка  $S_t = \{\alpha_i\}_{i=1}^t$ , то есть ставится проблема поиска лучшего решения на основе *имеющейся* информации. Если для восстановления грамматики перечислением наличие информатора крайне желательно, то при грамматическом выводе чаще ограничиваются текстуальным представлением. Мы также рассмотрим задачу восстановления по положительному образцу  $S_t^+$ .

Однако является ли совместимость с образцом языка удовлетворительным критерием при восстановлении грамматики? Для текстуального представления имеем два очевидных тривиальных решения: *ad hoc* и «беспорядочную» грамматику. Первая грамматика допускает только данные цепочки и содержит  $t$  правил  $\{S \rightarrow \alpha_i\}_{i=1}^t$ , вторая допускает вообще все цепочки (например, содержит правила вида  $\{S \rightarrow a_i S \mid a_i \in V_T\}; S \rightarrow \Lambda$ ).



Если бы требовалось просто построить грамматику, которая порождала данные цепочки, то эта задача решалась бы тривиально. Но такие решения нас интуитивно не устраивают. Что же мы хотим от грамматического вывода?

Обратим внимание, что предложенные тривиальные решения обладают следующими особенностями. Первое решение дает грамматику, не порождающую возможные отрицательные предложения, но при этом ни один новый положительный пример, не вошедший в образец, также порождаться не будет. Второе решение дает грамматику, порождающую любой новый положительный пример, но также порождающую и любое отрицательное предложение.

Здесь в завуалированном виде присутствует проблема обобщения, которая характерна для всех задач обучения. Казалось бы, нам нужна любая грамматика, удовлетворяющая набору цепочек. Но зачем нужна эта грамматика? Произвольная грамматика, просто порождающая данные положительные цепочки и не порождающая отрицательные цепочки, будет бесполезна (если информационная последовательность не является полной). Естественно, хотелось бы, чтобы грамматика *предсказывала*, какие новые цепочки возможны, а какие не должны порождаться. В приведенных тривиальных грамматиках либо обобщение отсутствует, либо выполнено чрезмерное обобщение, и, как следствие, теряется предсказательная сила.

Рассмотрим типичный эвристический метод грамматического вывода на следующем классическом примере положительного образца:

$$S_7 = \{caaab, bbaab, caab, bbab, cab, bbb, cb\}.$$

**На первом шаге** формируются нетерминальные символы и правила таким образом, чтобы они порождали исходные цепочки. Для первой цепочки можно ввести такие правила:

$$caaab: S \rightarrow cA_1, A_1 \rightarrow aA_2, A_2 \rightarrow aA_3, A_3 \rightarrow aA_4, A_4 \rightarrow b.$$

На основе приведенных правил может быть сформирована цепочка  $caaabb$  и только она. Аналогичным образом вводятся правила подстановки для следующих цепочек. Для второй цепочки это будут правила:

$$bbaab: S \rightarrow bA_5, A_5 \rightarrow bA_6, A_6 \rightarrow aA_7, A_7 \rightarrow aA_7, A_8 \rightarrow b.$$

Для третьей цепочки первое правило должно было бы иметь вид  $S \rightarrow cA_7$ , но уже на первом шаге эвристического алгоритма может производиться упрощение грамматики. Это упрощение заключается в том, что вместо введения нового нетерминального символа и нового правила подстановки используется уже введенный символ  $A_1$  с соответствующим правилом  $S \rightarrow cA_1$ . Тогда для порождения третьей цепочки потребуются правила:

$$caab: S \rightarrow cA_1, A_1 \rightarrow aA_2, A_2 \rightarrow aA_3, A_3 \rightarrow b,$$

из которых новым будет только правило  $A_3 \rightarrow b$ .

Для оставшихся цепочек формируются следующие правила:

$$bbab : S \rightarrow bA_5, A_5 \rightarrow bA_6, A_6 \rightarrow aA_7, A_7 \rightarrow b;$$

$$cab : S \rightarrow cA_1, A_1 \rightarrow aA_2, A_2 \rightarrow b;$$

$$bbb : S \rightarrow bA_5, A_5 \rightarrow bA_6, A_6 \rightarrow b;$$

$$cb : S \rightarrow cA_1, A_1 \rightarrow b.$$

С учетом повторяющихся правил получаем следующую систему правил:

$$S \rightarrow cA_1, S \rightarrow bA_5 \quad A_1 \rightarrow aA_2, A_1 \rightarrow b \quad A_2 \rightarrow aA_3, A_2 \rightarrow b$$

$$A_3 \rightarrow aA_4, A_3 \rightarrow b \quad A_4 \rightarrow b \quad A_5 \rightarrow bA_6$$

$$A_6 \rightarrow aA_7, A_6 \rightarrow b \quad A_7 \rightarrow aA_7, A_7 \rightarrow b \quad A_8 \rightarrow b$$

**На втором шаге** алгоритма осуществляется редукция (упрощение) сложной грамматики на основе некоторых эвристических правил. Рассмотрим такое правило: объединить (отождествить) такие нетерминальные символы  $A$  и  $B$ , которые входят в правила подстановки вида  $A \rightarrow \alpha$ ,  $B \rightarrow \alpha$ . Под объединением символов  $A$  и  $B$  понимается такая процедура, при которой вводится новый нетерминальный символ  $C$ , и все вхождения символов  $A$  и  $B$  заменяются этим символом. После объединения оба правила  $A \rightarrow \alpha$ ,  $B \rightarrow \alpha$  приобретают вид  $C \rightarrow \alpha$ , то есть число правил уменьшается (возможно также, что и некоторые другие правила после замены каждого из символов  $A$  и  $B$  на символ  $C$  будут тождественными).

Поскольку в построенной грамматике имеются правила подстановки  $A_1 \rightarrow b, A_2 \rightarrow b, A_3 \rightarrow b, A_4 \rightarrow b, A_6 \rightarrow b, A_7 \rightarrow b, A_8 \rightarrow b$ , символы  $A_1, A_2, A_3, A_4, A_6, A_7, A_8$  объединяются. Заменяем все их вхождения символом  $C$ . Обычно в процессе грамматического вывода объединение символов осуществляется попарно, но здесь для сокращения записи мы воспользуемся объединением сразу семи символов. После такого объединения в грамматике останутся следующие правила:

$$S \rightarrow cC, S \rightarrow bA_5 \quad C \rightarrow aC, C \rightarrow b \quad A_5 \rightarrow bC.$$

В данном случае полученный результат является окончательным, но вполне могло оказаться так, что объединение некоторых нетерминальных символов привело бы к возможности объединения новых нетерминальных символов, и процесс редукции грамматики продолжился бы.

Рассмотренный грамматический вывод начинается с *ad hoc* грамматики, совместимой с заданным образцом и порождающей наиболее узкий язык, содержащий только предложения из образца. На каждом шаге редукции грамматики порождаемый язык расширяется (или, по крайней мере, не сужается), то есть происходит постепенное обобщение на основе образца языка. В случае попарного объединения нетерминальных символов смысл этого обобщения прост: классы символов или грамматические категории, которые обозначаются некоторыми нетерминальными символами, объединяются с целью формирования более общих классов и категорий. При этом в процессе вывода текущая грамматика все время остается совместимой с образцом. Эти моменты (использование *ad hoc* грамматики в качестве нулевого приближения и ее постепенное упрощение с сохранением совместимости с образцом) характерны для большинства эвристических алгоритмов грамматического вывода. Если на каждом шаге вывода производится обобщение, то возникает вопрос в правомерности такого обобщения при выполнении того или иного эвристического условия. Этот вопрос, в конечном счете, сводится к выработке корректного критерия в машинном обучении.

Еще одним аспектом эвристических методов грамматического вывода, требующим детального рассмотрения, является выбор операций по преобразованию грамматики в процессе ее редукции. Мы рассмотрели пример одной такой операции – операции по объединению двух нетерминальных символов при выполнении достаточно жесткого условия (если есть правила вида  $A \rightarrow \alpha$ ,  $B \rightarrow \alpha$ , то символы  $A$  и  $B$  следует объединить). Могут быть использованы и другие эвристические правила. Например, применяется такая эвристика: если есть правила  $A \rightarrow ab$ ,  $B \rightarrow aC$  и  $C \rightarrow b$ , то нетерминальный символ  $A$  может быть заменен символом  $C$ , а правило  $A \rightarrow ab$  удалено. Использование разных наборов эвристик приводит к тому, что пути грамматического вывода оказываются различными. При этом нет гарантии, что разные пути вывода приведут к одному и тому же конечному результату – как и в случае градиентного спуска, алгоритм вывода может «застрять» в локальном минимуме. Выбор ограниченного числа эвристических приемов оказывается ненадежным.



Также обратим внимание на следующее. Исходная грамматика, строящаяся в приведенном примере, является очень частным видом грамматик, а именно автоматной грамматикой. При применении любого правила объединения символов тип грамматики не меняется (если быть более точным, то грамматика может из нерекурсивной превратиться в рекурсивную). Возникает необходимость введения таких операций по преобразованию грамматик, которые бы позволяли расширять тип грамматик (или приводили бы к правилам подстановки нового вида). Такой является, например, операция конструирования. При выполнении этой операции на основе правил вида  $A \rightarrow \alpha B$  и  $B \rightarrow \beta C$  формируется правило вида  $A \rightarrow \alpha \beta C$ ; старые правила удаляются только в том случае, если при порождении образца языка они всегда применялись вместе.

Еще одна операция, приводящая к изменению вида правил подстановки, – это операция по удалению нетерминального символа: если символ  $A$  входит в левую часть только одного правила подстановки вида  $A \rightarrow \alpha$ , то все вхождения этого символа в других правилах могут быть заменены цепочкой  $\alpha$ , а правило  $A \rightarrow \alpha$  удалено. В нашем примере удаление символа  $A_5$  приведет к системе правил  $S \rightarrow cC, S \rightarrow bbC, C \rightarrow aC, C \rightarrow b$ . При применении операций конструирования и удаления порождаемый язык не расширяется. При этом операция конструирования позволяет уменьшить сложность вывода, в то время как операция удаления – внутреннюю сложность грамматики, что еще раз ставит вопрос о введении адекватного критерия качества грамматик в задаче грамматического вывода. Приведенных операций все еще не хватает, чтобы выйти за рамки КС-грамматик (из-за этого и из-за сложности задачи синтаксического разбора для НС-грамматик при решении задачи грамматического вывода часто ограничиваются восстановлением КС-грамматик).

