Язык и порождающие грамматики

РГ	КС	КЗ	Тип 0
S → 0A A → 0A | 1B | 1 B → 1B | 1	S → AB A → 0A | 0 B → 1B | 1	S → AB A → 0A | 0 0A → 00A B → 1B | 1	S → 0AB1 A → 0A | ε 0A → 00A B → 1B | ε

Лемма о накачке для регулярных языков

Если задан регулярный язык L, то существует константа $n > 0$, такая, что если $\alpha \in L$ и $|\alpha| \geq n$, то цепочку α можно записать в виде $\alpha = \beta\gamma\delta$, где $0 < |\gamma|$, и тогда $\alpha = \beta\gamma^i\delta$, $\alpha \in L$ для всех $i > 0$.

Язык $L = \{0^m1^n \mid n, m > 0\}$ – регулярный язык

00^k1111

001111 000000001111 0000000000000001111

Лемма о накачке для регулярных языков (2)

Используя лемму о разрастании регулярных языков, докажем, что язык $L = \{a^n b^n \mid n > 0\}$ не является регулярным.

Положим, что $L = \{a^n b^n \mid n > 0\}$ – регулярный язык. Тогда должны существовать три подцепочки β, γ и δ, причем $\gamma \neq \varepsilon$, и $\beta\gamma^n\delta \in L$ для всех $n > 0$. Существуют три возможных варианта подцепочки γ:

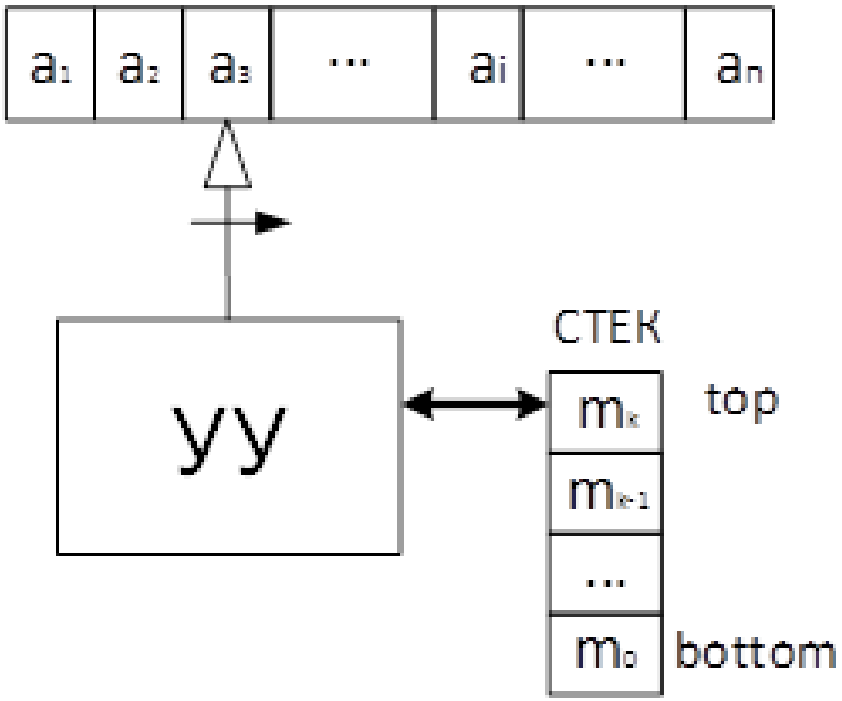
- γ состоит только из символов a (хотя бы одного a). Тогда цепочка $\beta\gamma^2\delta$ будет содержать больше символов a, чем b, и, значит, эта цепочка не принадлежит $\{ a^n b^n \mid n>0\}$; aabb aaaaabb
- γ состоит только из символов b (хотя бы одного b). Тогда цепочка $\beta\gamma^2\delta$ будет содержать больше символов b, чем a, и, значит, эта цепочка не принадлежит $\{ a^n b^n \mid n>0\}$; aaabbb aabbbbb
- $\gamma = a^i b^j$ $(i, j > 0)$, тогда даже цепочка $\beta\gamma^2\delta$ будет содержать символы b за которыми следуют символы a, что для языка $L = \{a^n b^n \mid n > 0\}$ недопустимо aaabbb aaabbabbabbb

Свойства регулярных языков

- Регулярные языки, а язык – это множество, замкнуты относительно следующих операций:
- пересечения;
- объединения;
- дополнения;
- разности;
- обращения;
- итерации;
- конкатенации;

Контекстно свободные языки и МП-автоматы

Структурная схема МП-автомата



- PDA = (X, S, s0, F, δ, M, m0), где
- X – множество входных символов (входной алфавит);
- S – конечное множество состояний МП-автомата;
- s_0 – начальное состояние МП-автомата ($s_0 \in S$);
- F – множество допускающих состояний ($F \subseteq S$);
- M – множество символов стека;
- m_0 – начальный символ стека ($m_0 \in M$), в начале работы стек содержит этот единственный символ;
- δ – функция переходов,

$\delta: (S \times (X \cup \{\varepsilon\}) \times M) \rightarrow (S \times M^*)$.

МП-автомат

- Начальная конфигурация МП-автомата определяется тройкой (s_0, ω, m_0);
- отношение непосредственной смены конфигураций, обозначаемое $\vdash$, пусть $\delta(s_1, t, m)$, причем $s_1 \in S$, $t \in X$, $m \in M$, содержит пару (s_2, α) $s_2 \in S$, $\alpha \in M^*$, тогда для всех цепочек $\omega \in X^*$ и $\beta \in M^*$ справедливо следующее отношение: $(s_1, t\omega, m\beta) \vdash (s_2, \omega, \alpha\beta)$
- $(s_1, t, m) \vdash (s_2, \alpha)$
- Конечной конфигурацией МП-автомата является тройка (q, ε, α) где $q \in F$, $\alpha \in M^*$
- путь по конфигурациям $(s_0, \omega, m_0) \vdash * (q, \varepsilon, \alpha)$ для некоторых $q \in F$ и $\alpha \in M^*$

Упрощение КС грамматик

- Пусть G = (VT, VN, P, S) – КС-грамматика.
- Нетерминальный символ A ∈ VN называется непроизводящим, если множество терминальных цепочек, выводимых из него, пусто, обозначается $\{\alpha \mid \alpha \in VT^*, A \Rightarrow* \alpha\} = \emptyset$.
- Символ, принадлежащий словарю грамматики V, называется недостижимым, если он не может быть получен ни в одной из сентенциальных форм.
- Непроизводящие и недостижимые символы называются *бесполезными*.
- Грамматика называется *приведенной*, если из неё удалены все правила, содержащие бесполезные символы.

Удаление непроизводящих символов

G = ({a, b}, {A, B, C, D, E, F, S}, P, S).

Исходное множество правил P	Формирование множества VN’	Результирующая грамматика
S → AC | BS |B **A → aA | aF** **B → CF | b** **C → cC | D** **D → aD | BD | C** **E → aA | BSA** **F → bB | b**	$VN'_0 = \{\emptyset\}$ $VN'_1 = \{B, F\}$ $VN'_2 = \{B, F, A, S\}$ $VN'_3 = \{B, F, A, S, E\}$ $VN'_4 = \{B, F, A, S, E\}$ $VN'_4 = VN'_3$ FINISH	G’ = (VT, VN’, P’, S) VT = {a, b} VN’ = {A, B, E, F, S} P’ = {S → BS | B A → aA | aF B → b E → aA | BSA F → bB | b}

Удаление недостижимых символов

Исходное множество правил	Формирование множеств VT’ и VN’	Результирующая грамматика
S → BS | B	$VN'_0 = \{S\}$ $VT'_0 = \emptyset$	
A → aA | aF	$VN'_1 = \{S, B\}$ $VT'_1 = \{b\}$	G’ = (VT’, VN’, P’, S)
B → b	$VN'_2 = \{S, B\}$ $VT'_2 = \{b\}$	VT’ = {b}
E → aA I BSA	$VN'_2 = VN'_1$	VN’ = {A, B, E, F, S}
F → bB | b	FINISH	P’ = {S → BS | B
		B → b}

Удаление ε-правил

- Контекстно-свободная грамматика G = (VT, VN, P, S) называется расширенной КС грамматикой, если в множестве правил P содержится одно или более ε-правила (правила вида A → ε).
- Грамматика называется S-расширенной КС грамматикой (VT, VN, P, S), если множество P содержит правило S → ε.
- Для любой расширенной КС грамматики G, такой что ε ∉ L(G), существует эквивалентная КС грамматика G’ без ε-правил.
- Для любой расширенной КС грамматики G, такой что ε ∈ L(G), существует эквивалентная S-расширенная грамматика G’.

Удаление ε-правил (2)

- Для заданной КС грамматики G нетерминальный символ A называется nullable, если A ⇒* ε, иначе говоря, из A выводима пустая цепочка. Множество Nullable для грамматики G включает в себя все нетерминалы, из которых выводимы пустые цепочки.

{S → ACA

A → aAa | B | C

B → bB | b

C → cC | ε}

МножествоNullable

Ø
{C}
{A, C}
{A, C, S}
{A, C, S}

Удаление ε-правил (3)

S → ACA

A → aAa | B | C

B → bB | b

C → cC

Исходный набор правил P’

S → ε | ACA

A → aAa | B | C

B → bB | b

C → cC

Результирующий набор правил P’

S → ε | ACA | AA | AC | CA | A | C

A → aAa | aa | B | C

B → bB | b

C → cC | c

Удаление цепных правил.

Рассмотрим следующие правила

$A \to aA \mid a \mid B$

$B \to bB \mid b \mid C$

Цепное правило

$A \to B$

Замена правой части правила $A \to B$ на все возможные правые части правил с B в левой части правила

$A \to aA \mid a \mid bB \mid b \mid C$

$B \to bB \mid b \mid C$

Удаление цепных правил (2)

Действие	Множество P’
P’ = P - {(A → α) ∈ P	S → ACA | AA | AC | CA
для всех $\alpha \notin$ VN}	A → aAa | aa
	B → bB | b
	C → cC | c
Chain(S) = {S, A, C, B}	S → ACA | AA | AC | CA | aAa | aa | bB | b | cC | c
Добавляем правила	A → aAa | aa
S → aAa | aa	B → bB | b
S → bB | b	C → cC | c
S → cC | c	
Chain(A) = {A, C, B}	S → ACA | AA | AC | CA | aAa | aa | bB | b | cC | c
Добавляем правила	A → aAa | aa | bB | b | cC | c
A → bB | b	B → bB | b
A → cC | c	C → cC | c
Chain(B) = {B}	Без изменений
Chain(C) = {C}	Без изменений

Удаление цепных правил (3).

- Было

S → ACA | AA | AC | CA | A | C

A → aAa | aa | B | C

B → bB | b

C → cC | c }

- Стало

S → ACA | AA | AC | CA | aAa | aa | bB | b | cC | c

A → aAa | aa | bB | b | cC | c

B → bB | b

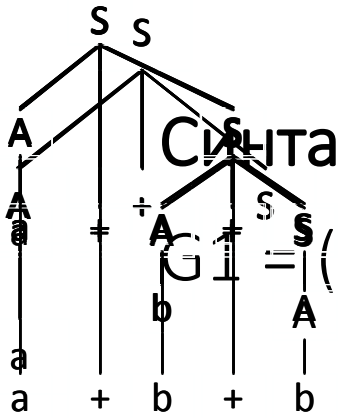
C → cC | c

Синтаксические деревья

- Процесс вывода сентенциальной формы для КС-грамматики G = (VT, VN, P, S) может быть наглядно представлен с помощью *синтаксического дерева*, или иначе дерева разбора. Синтаксическое дерево, соответствующее сентенциальной форме, это связный ацикличный граф такой, что:
- корень дерева – вершина с нулевой степенью захода – помечается начальным символом грамматики S;

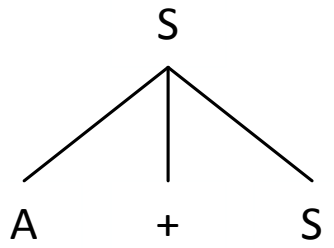
Синтаксические деревья

- узлы дерева помечаются нетерминальными символами грамматики; если узел дерева помечен символом A, $A \in VN$, и все исходящие из этого узла дуги связаны с вершинами, помеченными $v_1, v_2, \ldots v_n$, причем $n>0$; $v_i \in VN \cup VT \cup \{\varepsilon\}$; то в грамматике должно существовать правило $A \rightarrow v_1 v_2 \ldots v_n \in P$.
- листья дерева (терминальные узлы) для общего случая сентенциальной формы помечаются символами из множества $VN \cup VT \cup \{\varepsilon\}$; если сентенциальная форма является предложением языка, то терминальные узлы помечаются символами из множества $VT \cup \{\varepsilon\}$.
- При обходе листьев дерева слева направо получается цепочка символов, соответствующая сентенциальной форме, вывод которой иллюстрирует синтаксическое дерево.

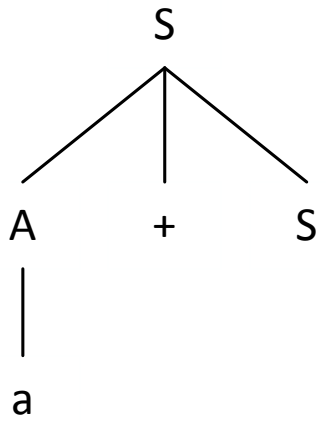


Синтаксические деревья

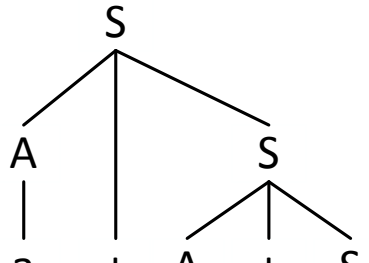
$G_1 = (\{a, b, +\}, \{S, A\}, \{S \to A \mid A+S;\ A \to a \mid b\}, S)$. a+b+b



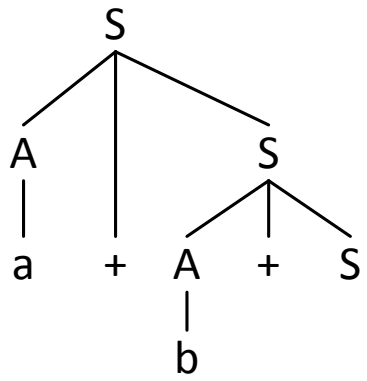
$S \Rightarrow A + S$



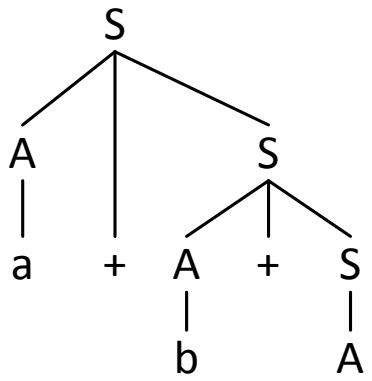
$S \Rightarrow_+ a + S$



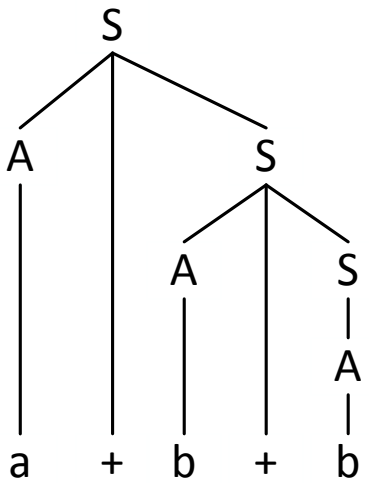
$S \Rightarrow_+ a + A + S$



$S \Rightarrow_+ a + b + S$



$S \Rightarrow_+ a + b + A$



$S \Rightarrow_+ a + b + b$