Базовые понятия теории формальных языков

Определение 1. *Формальный язык — это множество $\mathcal{M}$ слов над алфавитом Σ (обозначается $\mathcal{M} \subseteq \Sigma^*$, здесь знак * — итерация Клини). Обычно подразумевает наличие формальных правил, определяющих корректность формы (т.е. синтаксиса) слов из $\mathcal{M}$.*

Определение 2. *Формальный язык — это категория (т.е. способ задания объектов и их взаимосвязей в форме направленного графа). Отличие от теоретико-множественного подхода — способ описания не синтаксиса слов, а отношения между ними.*

Классический пример: A man saw a dog with a telescope.

Предложение 1.
- *Язык $\mathcal{M}$ разрешимый $\Leftrightarrow$ для любого слова w существует алгоритм проверки принадлежности w к $\mathcal{M}$ (всегда завершающийся и дающий точный, либо положительный, либо отрицательный ответ).*
- *Язык $\mathcal{M}$ перечислимый $\Leftrightarrow$ для любого слова w существует алгоритм, положительно отвечающий на вопрос принадлежности w к $\mathcal{M}$ за конечное время (но, возможно, зацикливающийся, если $w \notin \mathcal{M}$).*

Перечислимый, но не разрешимый: язык программ, завершающихся на входе 0 (на любом достаточно мощном ЯП).

Далее разрешимые языки можно классифицировать по минимально необходимой сложности разрешающего алгоритма (P-разрешимые, $ExpTime$-разрешимые...)

Примеры формальных языков

- $\{\underbrace{aa...a}_{n\text{раз}}\ \underbrace{bb...b}_{n\cdot 3\text{раз}}\}$ (сокращаем до $\{a^n b^{3n}\}$);
- палиндромы чётной длины в русском языке;
- правильно записанные арифметические выражения с $\cdot$, $+$ над натуральными числами;
- правильные скобочные последовательности;
- язык тождественно истинных формул логики предикатов;
- язык правильно типизированных программ на ЯП со статическими типами;
- язык, описывающий все разрешимые за линейное время формальные языки.

Представления формальных языков

- Свёртки множеств
- Системы переписывания термов
- Распознающие / порождающие машины
- Алгебраические выражения
- Алгебраические структуры
- Формулы логики предикатов

Пример представления

Язык слов в алфавите $\{a, b\}$ с чётным количеством букв a.

- Свёртка: $\{w \in \{a, b\}^* \mid 2 \text{ делит } |w|_a\}$. $|w|_t$ — количество вхождений терма t в слово w.
- Система переписывания термов:

$$\begin{array}{lll} S \to "a" ++ T & S \to "b" ++ S & S \to \varepsilon \\ T \to "a" ++ S & T \to "b" ++ T & \end{array}$$

  А можно и по-другому:

$$\begin{array}{ll} S \to "a" ++ S ++ "a" & S \to "b" ++ S \\ S \to S ++ "b" & S \to \varepsilon \end{array}$$

  Здесь и далее ε — стандартное обозначения для пустого слова (строки нулевой длины). Поскольку структура данных — слова, то кавычки и знак конкатенации $++$ дальше опускаются.

- Распознающие машины:

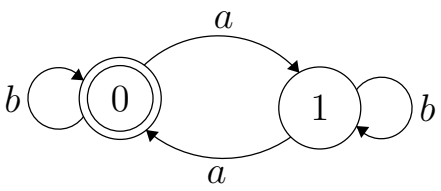


Рисунок 1 — К параграфу «Пример представления»

А можно иначе:

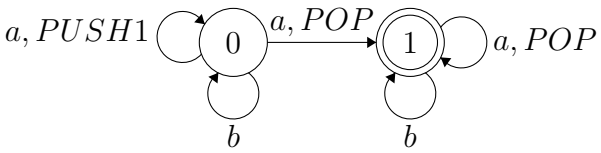


Рисунок 2 — К параграфу «Пример представления»

- Алгебраические выражения:

  $(b^*ab^*ab^*)^*$

  А можно и так:

  $(b^*ab^*a)^*b^*$

- Алгебраические структуры:

  Класс эквивалентности слова ε в полугруппе с соотношениями $aa \to \varepsilon$, $b \to \varepsilon$.

- Формулы логики предикатов: без введения считающих предикатов не выразима в логике предикатов первого порядка (но выразима в логике одноместных предикатов второго порядка).

Анализ свойств языков

Проверить, действительно ли данная система переписывания термов порождает язык $\{w \,|\, |w|_a$ делится на $2\}$, если начальным состоянием является S.

$$S \to a\,S\,a \quad S \to b\,S \quad S \to S\,b \quad S \to \varepsilon$$

- Необходимо доказать, что все указанные слова порождаются системой (например, по индукции).
- А также что никакие другие не порождаются.

Предположим, что система порождает слова с нечётным числом букв a. Выберем из них такое, которое выводится из S за самое малое число шагов. Покажем, что каким бы ни был предпоследний шаг вывода, его можно поменять на $S \to \varepsilon$ и получится слово с нечётным числом букв a, вывод которого ещё короче.

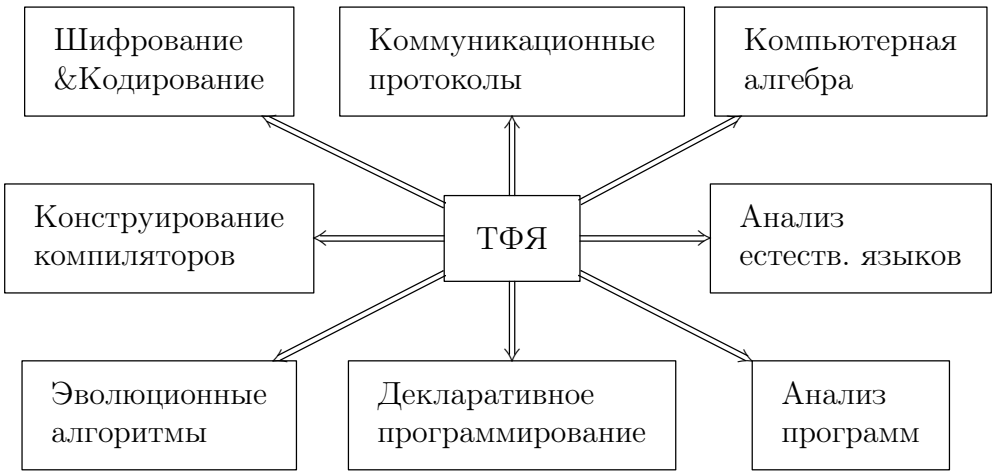


Рисунок 3 — Области применения

Структура курса

- Два рубежных контроля × 15 баллов.
- Пять лабораторных работ × 8 баллов.
  - Java, Python, Go, JS — без бонуса
  - C/C++, Kotlin, TypeScript — бонус +1 балл
  - Rust, Dart, все лиспы, Scala — бонус +2 балла
  - Lua, Haskell, Erlang, Рефал — бонус +3 балла
  - Agda, Idris (с доказательствами) — 5 баллов за курс
- С момента выдачи лабораторной работы:
  - 0-14 дней — сдача за полный балл
  - 15-21 день — сдача со штрафом -1 балл
  - 22-28 дней — сдача со штрафом -2 балла
  - 29-∞ — сдача со штрафом -3 балла

Определение 3. *Сигнатура — множество пар $\langle f, n\rangle$ из имени конструктора f и его местности n.*

Определение 4. *Пусть V — множество переменных, F — множество конструкторов; множество термов $T(F)$ над F определяется рекурсивно:*

- *все элементы V — термы;*
- *если $\langle f, n\rangle$ — конструктор и $t_1, \ldots, t_n$ — термы, то $f(t_1, \ldots, t_n)$ — терм;*
- *других термов нет.*

Term Rewriting Systems

Пусть V — множество переменных, F — множество конструкторов (сигнатура); $T(F)$ — множество термов над множеством конструкторов F. TRS — набор правил переписывания вида $\Phi_i \to \Psi_i$, где Φ_i, Ψ_i — термы в $T(F)$. Правило переписывания $\Phi_i \to \Psi_i$ применимо к терму t, если t содержит подтерм, который можно сопоставить (унифицировать) с Φ_i.

Если к терму t не применимо ни одно правило переписывания TRS, терм называется нормализованным.

Имея правила переписывания вида $f(g(x)) \to g(g(f(x)))$ и $g(g(x)) \to f(x)$, каждое из них можно применить к терму $f(g(g(f(g(f(g(g(g(Z)))))))))$ тремя разными способами.

Определение 5. *TRS называется конфлюэнтной, если для любых двух термов t, s, которые получаются переписыванием одного и того же терма u, существует терм v такой, что t, s оба переписываются в v.*

Формальнo:

$\forall u, t, s(u \to^* t \;\&\; u \to^* s \Rightarrow \exists v(t \to^* v \;\&\; s \to^* v))$

Конфлюэнтные системы поддаются распараллеливанию и легко оптимизируются.

$\to$ — переписывание за 1 шаг;

$\to^*$ — переписывание за произвольное число шагов, начиная с 0.

Особенности TRS

- Недетерминированные.
- Нет ограничений на порядок применения правил.
- Не обязательно конфлюэнтны.
- Могут порождать бесконечные цепочки.

Пример 1. $f(x, x) \to a$
$f(x, g(x)) \to b$
$c \to g(c)$
Терм, где нарушается конфлюэнтность?

Пример 2. $A \to CA$
$Cz \to Dz(Cz)$
$Dzz \to E$
Способы преобразовать A?

Пример 3.
- *TRS 1:*

$f(0, 1, x) \to f(x, x, x)$

- *TRS 2:*

  $g(x, y) \to x$

  $g(x, y) \to y$

Как можно вычислить $f(g(0,1), g(0,1), g(0,1))$*?*

Определение 6. *Частичный порядок* $\preceq$ *является фундированным (wfo) на множестве* M*, если в* M *не существует бесконечных нисходящих цепочек относительно* $\preceq$ *(говоря о множестве термов, иногда такой* $\preceq$ *называют нётеровым).*

Частичный порядок $\preceq$ *является монотонным в алгебре* A*, если* $\forall f, t_1, ..., t_n, s, s'(s \preceq s' \Rightarrow f(t_1, \ldots, s, \ldots$ $f(t_1, \ldots, s', \ldots, t_n))$ *(строго монотонным, если при этом неверно обратное).*

Определение 7. *Фундированная монотонная алгебра (ФуМА) над множеством функциональных символов* F *— это фундированное множество* $\langle A, > \rangle$ *такое, что для каждого функционального символа* $f \in F$ *существует функция* $f_A : A^n \to A$*, строго монотонная по каждому из аргументов.*

Определим расширение произвольного отображения σ из множества переменных в A следующим образом:

- $[x, \sigma] = \sigma(x)$;
- $[f(t_1, \ldots, t_n), \sigma] = f_A([t_1, \sigma], \ldots, [t_n, \sigma])$.

Предложение 2. *TRS* $\{l_i \to r_i\}$ *совместна с ФуМА* $A \Leftrightarrow$ *для всех* i *и для всех* σ *выполняется условие* $[l_i, \sigma] > [r_i, \sigma]$.

Теорема 1. *TRS не порождает бесконечных вычислений (завершается), если и только если существует совместная с ней ФуМА.*

Стандартные способы определения f_A:

- лексикографический порядок на множестве имён F + отношение подтерма;
- построение монотонно возрастающей (по каждому аргументу) числовой функции, соответствующей f_A.

Оба случая подразумевают, что в построенной модели целое больше части, т.е. всегда выполняется $f(t) > t$.

Определение 8. $f(t_1, \ldots, t_n) >_{lo} g(u_1, \ldots, u_m)$ *(этот порядок также называют порядком Кнута–Бендикса) если и только если выполнено одно из условий:*

1. $\exists i(1 \le i \le n \;\&\; t_i = g(u_1, \ldots, u_m))$;
2. $\exists i(1 \le i \le n \;\&\; t_i >_{lo} g(u_1, \ldots, u_m))$;
3. $(f > g) \;\&\; \forall i(1 \le i \le m \Rightarrow f(t_1, \ldots, t_n) >_{lo} u_i)$;
4. $(f = g) \;\&\; \forall i(1 \le i \le n \Rightarrow f(t_1, \ldots, t_n) >_{lo} u_i)$ *и n-ка* $(t_1, \ldots, t_n)$ *лексикографически больше, чем* $(u_1, \ldots, u_n)$ *(т.е. первый её не совпадающий с* u_i *элемент* t_i *удовлетворяет условию* $t_i >_{lo} u_i$*).*

Пример 4. *Проверить завершаемость TRS методом* $>_{lo}$*:*

$f(g(x)) \to g(h(x, x))$

$g(f(x)) \to h(g(x), x)$

- Первое правило переписывания вынуждает либо $g(x) >_{lo} g(h(x,x))$ (по условию 1 или 2) — что невозможно, потому что x должно лексикографически оказаться больше $h(x,x)$ (по условию 4); либо $f > g$ и $f(g(x)) > h(x,x)$ (по условию 3). В этом случае можно взять также $f > h$. Неравенство $f(g(x)) > x$ выполняется тривиально.

- Второе правило переписывания удовлетворяет условию завершаемости по условию 2, например, если показать, что $f(x) >_{lo} h(g(x), x)$. Уже имеем $f > h$, поэтому достаточно показать $f(x) >_{lo} g(x)$ и $f(x) >_{lo} x$. Оба условия тривиально выполняются из допущений выше.

Пример 5. *Проверить завершаемость TRS методом построения монотонной функции:*
$f(g(x,y)) \to g(h(y), x)$
$h(f(x)) \to f(x)$.

- Завершаемость по второму правилу переписывания автоматически выполняется по свойству подтерма. Поэтому то, что функция f стоит на двух его сторонах, не дает никаких указаний относительно того, стоит ли делать f_A быстро растущей или медленно. Все подсказки содержатся только в первом правиле переписывания.

- По первому правилу переписывания видно, что f_A надо делать большой (f стоит только слева), а h_A нет (h есть только справа). Положим $f_A(x) = 10 \cdot (x+1)$, $h_A(x) = x + 1$. Тогда должно выполняться $10 \cdot (g_A(x,y) + 1) > g_A(y+1, x)$. Этому неравенству удовлетворяет, например, $g_A(x,y) = x + y$.

Общие комментарии

- Не обязательно добиваться выполнения неравенства на образах f_A на всём множестве $\mathbb{N}$. Поскольку любой отрезок $\mathbb{N}$ от k и до бесконечности фундирован, а все образы f_A монотонны, они замкнуты на этом отрезке. Поэтому, если неравенство не выполняется для нескольких первых чисел натурального ряда, этим можно пренебречь.

- Если не получается применить $>_{lo}$ или подобрать числовую функцию, это ещё не значит, что TRS не завершается. См. пример Зантемы: $f(g(x)) \to g(f(f(x)))$.

Предложение 3. *Если TRS определена над алфавитом Σ, а нас интересует порождаемый ею язык в $\Sigma' \subset \Sigma$, то элементы Σ' обычно называются терминалами, а элементы $\Sigma \setminus \Sigma'$ — нетерминалами.*

В этом случае значащие (порождающие) нетерминалы обязательно должны встречаться хотя бы в одной левой части правила переписывания (иначе такой нетерминал не сможет быть переписан в слово над Σ').

Терминалы также могут встречаться в левых частях правил (это не так только для некоторых классов систем переписывания термов).

Определение 9. *Грамматика — это четвёрка $G = \langle N, \Sigma, P, S \rangle$, где:*

- N — *алфавит нетерминалов;*
- Σ — *алфавит терминалов;*
- P — *множество правил переписывания $\alpha \to \beta$ типа $\langle (N \cup \Sigma)^+ \times (N \cup \Sigma)^* \rangle$;*
- $S \in N$ — *начальный символ.*

$\alpha \Rightarrow \beta$, если $\alpha = \gamma_1 \alpha' \gamma_2$, $\beta = \gamma_1 \beta' \gamma_2$, и $\alpha' \to \beta' \in P$.
$\Rightarrow^*$ — рефлексивное транзитивное замыкание $\Rightarrow$.

Определение 10. *Язык $L(G)$, порождаемый G — множество $\{u \mid u \in \Sigma^* \;\&\; S \Rightarrow^* u\}$.*

По-разному воспринимают переименовку:

- Переменные vs конструкторы в TRS;
- Нетерминалы vs терминалы в грамматиках.

Определение 11. *Для любой инъективной переименовки σ применение σ к правилам грамматики/trs для переменных и нетерминалов также называется α-преобразованием.*

- α-преобразование не меняет терминальный язык;
- обычно термы различаются с точностью до α-преобразования.

Неформально: контейнеры определяются не именем, а содержимым (см. экстенсиональность в логике).

$A, B \in N$, $a \in \Sigma^*$, $\alpha, \beta \in (N \cup \Sigma)^*$, $\gamma \in (N \cup \Sigma)^+$

Иерархия грамматик

Тип 0	Рекурсивно-перечислимые	$\forall$
Тип 1	Контекстно-зависимые	$\alpha A \beta \to \alpha \gamma \beta$, $\gamma \neq \varepsilon$
Тип 2	Контекстно-свободные	$A \to \gamma$
Тип 3	Праволинейные (регулярные)	$A \to a$, $A \to aB$

Пример 6.

Тип 0	$\{u \mid L(u) = L(r)\}$, *r — фикс. regex, u — regex;*
Тип 1	$\{ww \mid w \in \Sigma^+\}$
Тип 2	*непустые палиндромы в алфавите* $\{a, b\}$
Тип 3	$\{w \mid w = aw_1 \;\&\; (w_1 = a^{2k} \vee w = a^{3k} \vee w_1 \neq a^{5k})\}$

$A, B \in N$, $a \in \Sigma^*$, $\alpha, \beta \in (N \cup \Sigma)^*$, $\gamma \in (N \cup \Sigma)^+$.

Иерархия грамматик

Тип 0	Рекурсивно-перечислимые	$\forall$
Тип 1	Контекстно-зависимые	$\alpha A \beta \to \alpha \gamma \beta$, $\gamma \neq \varepsilon$
		$\vee\ S \to \varepsilon \;\&\; \forall p : \alpha \to \beta \in P \forall \beta_1, \beta_2 (\beta \neq \beta_1 S \beta_2)$
Тип 2	Контекстно-свободные	$A \to \alpha$
Тип 3	Регулярные	$A \to a$, $A \to aB$, $A \to \varepsilon$

Регулярные грамматики и выражения. Теорема Клини

Определение 12. *Грамматика — это четвёрка $G = \langle N, \Sigma, P, S \rangle$, где:*

- N — *алфавит нетерминалов;*
- Σ — *алфавит терминалов;*
- P — *множество правил переписывания $\alpha \to \beta$ типа $\langle (N \cup \Sigma)^+ \times (N \cup \Sigma)^* \rangle$;*
- $S \in N$ — *начальный символ.*

$\alpha \Rightarrow \beta$, если $\alpha = \gamma_1 \alpha' \gamma_2$, $\beta = \gamma_1 \beta' \gamma_2$, и $\alpha' \to \beta' \in P$.

$\Rightarrow^*$ — рефлексивное транзитивное замыкание $\Rightarrow$.

Грамматики

Язык $\mathscr{L}(G)$, порождаемый G — множество $\{u \mid u \in \Sigma^* \ \&\ S \Rightarrow^* u\}$. Сентенциальная форма — элемент множества $\{u \mid u \in (N \cup \Sigma)^* \ \&\ S \Rightarrow^* u\}$.

Регулярные грамматики и НКА

Регулярная грамматика имеет правила вида $S \to \varepsilon$ (причём S не встречается в правых частях никаких правил), $T_i \to a_i$, $T_i \to a_i\, T_j$.

НКА (неформально) определяется списком правил перехода и финальными состояниями.

- $T_i \to a_i\, T_j$ соответствует переходу $\langle T_i, a_i, T_j \rangle$;
- $T_i \to a_i$ соответствует переходу $\langle T_i, a_i, F \rangle$, где F — уникальное финальное состояние;
- $S \to \varepsilon$ соответствует объявлению S финальным.

Лемма о накачке

Рассмотрим слово $w \in \mathscr{L}(G)$, $|w| \geq n+1$. Оно получается применением не меньше, чем $n+1$ правил $\Rightarrow$ после применения хотя бы двух из них в сентенциальной форме справа будет стоять один и тот же нетерминал A.

Предложение 4. *Если G — регулярная, то существует такое $n \in \mathbb{N}$, что* $\forall w\big(w \in \mathscr{L}(G)\ \&\ |w| > n \Rightarrow \exists w_1, w_2, w_3\big(|w_2| > 0\ \&\ |w_1| + |w_2| \leq n\ \&\ w = w_1\ w_2\ w_3\ \&\ \forall k(k \geq 0 \Rightarrow w_1\ {w_2}^k\ w_3 \in \mathscr{L}(G))\big)\big)$.

$$S \twoheadrightarrow \cdots \twoheadrightarrow \Phi\ A \twoheadrightarrow \cdots \twoheadrightarrow \Phi\ \Psi\ A \twoheadrightarrow \cdots \twoheadrightarrow \Phi\ \Psi\ \Theta$$

Рисунок 4 — К параграфу «Лемма о накачке»

$$\overbrace{\qquad}^{\text{не больше } n \text{шагов}}$$
$$\Downarrow$$
$$|\Phi| + |\Psi| \leq n$$

Известно, что $|\Phi| + |\Psi| \leq n$.

$$\overbrace{\qquad}^{\rho_2\text{: вывод } \Psi A \text{из } A}$$

$$S \twoheadrightarrow \cdots \twoheadrightarrow \Phi\ A \twoheadrightarrow \cdots \twoheadrightarrow \Phi\ \Psi\ A \twoheadrightarrow \cdots \twoheadrightarrow \Phi\ \Psi\ \Theta$$

Рисунок 5 — К параграфу «Лемма о накачке»

$$\overbrace{\qquad}^{\rho_1\text{: вывод } \Phi A \text{из } S} \quad \overbrace{\qquad}^{\rho_3\text{: вывод } \Theta \text{из } A}$$

Все выводы вида $\rho_1\, (\rho_2)^*\, \rho_3$ допустимы в $G \Rightarrow \forall k(\Phi\ \Psi^k\ \Theta \in \mathscr{L}(G))$.

Примеры применения леммы о накачке

Обозначим обращение (reversal) слова w как w^R. Рассмотрим язык $\mathscr{L} = \{w\,w^R \mid w \in \Sigma^+\}$.

Пусть длина накачки — n. Рассмотрим слово $b^{n+1}a\,a\,b^{n+1} \in \mathscr{L}$. Поскольку $|\Phi| + |\Psi| \leq n$, то $\Psi = b^k$, $k \geq 1$. Но $b^m a\,a\,b^n \notin \mathscr{L}$, если $m \neq n$. Поэтому $\mathscr{L}$ — не регулярный.

Рассмотрим язык $\mathscr{L}' = \{a^n b^m \mid n \neq m\}$.

Пусть длина накачки — n. Рассмотрим множество слов $a^n b^{n+n!} \in \mathscr{L}'$. Поскольку $|\Phi| + |\Psi| \leq n$, то $\Psi = a^k$, $k \geq 1$. Но для всех $k \leq n$ $\exists v(n + k * v = n + n!)$. Поэтому слово вида $a^{n+n!}b^{n+n!} \in \mathscr{L}'$, что абсурдно. Следовательно, $\mathscr{L}'$ не является регулярным.

Нерегулярные языки

Пусть $\mathscr{L} = \{w \mid |w|_a = |w|_b\}$. Все слова вида $a^k b^k$ принадлежат $\mathscr{L}$. Пусть длина накачки равна n. Рассмотрим слово $a^n b^n$. Поскольку $|\Phi| + |\Psi| \leq n$, то $\Psi = a^k$, $k > 0$. Но слова $a^{n+k*i}b^n$ не принадлежат $\mathscr{L}$.

Совпадает ли $\mathscr{L}$ с языком правильных скобочных последовательностей P (язык Дика)? Если да, доказать. Если нет, исследовать язык $L \setminus P$. Регулярен ли он?

Гипотеза

G — регулярная $\overset{???}{\iff}$ существует такое $n \in \mathbb{N}$, что $\forall w\big(w \in \mathscr{L}(G)\ \&\ |w| > n \Rightarrow \exists w_1, w_2, w_3\big(|w_2| > 0\ \&\ |w_1| + |w_2| \leq n\ \&\ w = w_1\ w_2\ w_3\ \&\ \forall k(k \geq 0 \Rightarrow w_1\ {w_2}^k\ w_3 \in \mathscr{L}(G))\big)\big)$.

Анализ на достаточность

Рассмотрим язык $\mathscr{L} = \{w\,w^R\,z \mid w \in \Sigma^+\ \&\ z \in \Sigma^+\}$ и $n = 4$.

- Если $|w| = 1$, тогда можно разбить слово $w\,w^R\,z$ так: $\Phi = w\,w^R$, $\Psi = z[1]$, $\Theta = z\big[2..|z|\big]$. Тогда для всех k $\Phi\,\Psi^k\,\Theta \in \mathscr{L}$.
- Если $|w| \geq 2$, тогда разбиваем так: $\Phi = \varepsilon$, $\Psi = w[1]$, $\Theta = w\big[2..|w|\big]\,w^R\,z$. Слова $w\big[2..|w|\big]\,w^R\,z$ и $w[1]^k\,w\big[2..|w|\big]\,w^R\,z$ при $k \geq 2$ также принадлежат $\mathscr{L}$.

Смысл леммы о накачке

Структура доказательства указывает, что длина накачки n регулярного языка $\mathscr{L}$ не больше (возможно, меньше) числа нетерминалов в минимальной грамматике для $\mathscr{L}$.

Предложение 5. *Рассмотрим $\mathscr{L} = a \mid b \mid (a\,\{a \mid b\}^* a)|(b\,\{a \mid b\}^* b)$. Если выбрать $n = 2$, то в качестве Ψ можно взять вторую букву слова из $\mathscr{L}$. Пусть G имеет два нетерминала S, T и распознаёт $\mathscr{L}$. Если G содержит правила $S \to aT$ и $S \to bT$ (или $S \to aS$, $S \to bS$), то для некоторого непустого z слова вида az и bz будут либо оба принадлежать $\mathscr{L}$, либо нет, чего не может быть. Значит, G содержит либо пару $S \to aT$, $S \to bS$, либо пару $S \to bT$, $S \to aS$. Рассмотрим первый случай. Тогда для некоторого непустого z имеем $az \in \mathscr{L} \Leftrightarrow b^+az \in \mathscr{L}$, что абсурдно.*

Допустимые операции

- A^* — замыкание Клини — ноль или больше итераций A;
- A^+ — одна или больше итерация A;
- $A?$ — 0 или 1 вхождение A;
- $A \mid B$ — альтернатива (вхождение либо A, либо B).

Предложение 6. *Если r_1, r_2 — $\mathcal{RE}$, тогда*

- $r_1 \mid r_2$ — $\mathcal{RE}$;

- $r_1 r_2$ — $\mathcal{RE}$;
- r_1^*, r_2^+ — $\mathcal{RE}$.

Построение 1. *Дано: G_1 и G_2 — праволинейные. Построить $G : \mathscr{L}(G) = \mathscr{L}(G_1) \cup \mathscr{L}(G_2)$.*

1. *Переименовать нетерминалы из N_1 и N_2, чтобы стало $N_1 \cap N_2 = \varnothing$ (сделать α-преобразование). Применить переименовку к правилам G_1 и G_2.*
2. *Объявить стартовым символом свежий нетерминал S и для всех правил G_1 вида $S_1 \to \alpha$ и правил G_2 вида $S_2 \to \beta$, добавить правила $S \to \alpha$, $S \to \beta$ в правила G.*
3. *Добавить в правила G остальные правила из G_1 и G_2.*

Построение 2. *Дано: G_1 и G_2 — праволинейные. Построить $G : \mathscr{L}(G) = \mathscr{L}(G_1)\ \mathscr{L}(G_2)$.*

1. *Переименовать нетерминалы из N_1 и N_2, чтобы стало $N_1 \cap N_2 = \varnothing$ (сделать α-преобразование).*
2. *Построить из G_1 её вариант без ε-правил (см. ниже).*
3. *По всякому правилу из G_1 вида $A \to a$ строим правило G вида $A \to aS_2$, где S_2 — стартовый нетерминал G_2.*
4. *Добавить в правила G остальные правила из G_1 и G_2. Объявить S_1 стартовым.*
5. *Если $\varepsilon \in \mathscr{L}(G_1)$ (до шага 2), то по всем $S_2 \to \beta$ добавить правило $S_1 \to \beta$.*

Построение 3. *Дано: G_1 — праволинейная. Построить $G : \mathscr{L}(G) = \mathscr{L}(G_1)^+$.*

1. *Построить из G_1 её вариант без ε-правил.*
2. *По всякому правилу из G_1 вида $A \to a$ строим правило G вида $A \to aS_1$, где S_1 — стартовый нетерминал G_1.*
3. *Добавить в правила G все (включая вида $A \to a$) правила из G_1. Объявить S_1 стартовым.*
4. *Если $\varepsilon \in \mathscr{L}(G_1)$ (до шага 2), добавить правило $S_1 \to \varepsilon$ и вывести S_1 из рекурсии.*

Построение 4. *Дано: G — праволинейная. Построить G' без правил вида $A \to \varepsilon$ такую, что $\mathscr{L}(G') = \mathscr{L}(G)$ или $\mathscr{L}(G') \cup \{\varepsilon\} = \mathscr{L}(G)$.*

1. *Перенести в G' все правила G, не имеющие вид $A \to \varepsilon$.*
2. *Если существует правило $A \to \varepsilon$, то по всем правилам вида $B \to aA$ дополнительно строим правила $B \to a$.*

Построение 5. *Дано: G_1, G_2 — праволинейные. Построить G' такую, что $\mathscr{L}(G') = \mathscr{L}(G_1) \cap \mathscr{L}(G_2)$.*

1. *Построить стартовый символ G' — пару $\langle S_1, S_2 \rangle$, где S_i — стартовый символ грамматики G_i.*
2. *Поместить $\langle S_1, S_2 \rangle$ в множество U неразобранных нетерминалов. Множество T разобранных нетерминалов объявить пустым.*
3. *Для каждого очередного нетерминала $\langle A_1, A_2 \rangle \in U$:*
   (а) если $A_1 \to a \in G_1$, $A_2 \to a \in G_2$, тогда добавить в G' правило $\langle A_1, A_2 \rangle \to a$;

(b) *если* $A_1 \to aA_3 \in G_1$, $A_2 \to aA_4 \in G_2$, *тогда добавить в* G' *правило* $\langle A_1, A_2\rangle \to a\langle A_3, A_4\rangle$, *а в* U — *нетерминал* $\langle A_3, A_4\rangle$, *если его ещё нет в множестве* T;

(c) *если все пары правил, указанные выше, были обработаны, тогда переместить* $\langle A_1, A_2\rangle$ *из* U *в* T.

4. *Повторять шаг 3, пока множество* U *не пусто.*

5. *Если* $\varepsilon \in \mathscr{L}(G_1)$ & $\varepsilon \in \mathscr{L}(G_2)$, *тогда добавить в* G' *правило* $\langle S_1, S_2\rangle \to \varepsilon$.

Теорема 2. *Если* $E \in \mathcal{RE}$, *то существует праволинейная регулярная грамматика* G *такая, что* $\mathscr{L}(G) = \mathscr{L}(E)$.

Будем строить сразу же НКА, распознающий то же слово, что и E. Для этого определим следующие множества:

- $\mathrm{First}(E)$ — множество символов, с которых может начинаться слово, распознаваемое E.
- $\mathrm{Last}(E)$ — множество символов, которыми может заканчиваться слово, распознаваемое E.
- $\mathrm{Next}(E)$ — множество пар символов, которые могут идти в словах, распознаваемых E, друг за другом.

В E пронумеруем все символы из Σ разными номерами. Для полученного E' построим $\mathrm{First}(E')$, $\mathrm{Last}(E')$, $\mathrm{Next}(E')$.

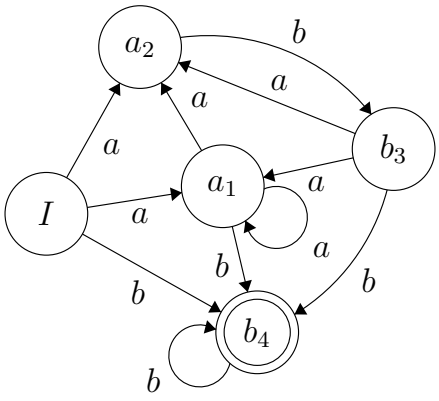
- Введём состояния, соответствующие буквам E' (нумерованным), а также входное состояние I.
- Если $\tau \in \mathrm{First}(E')$, тогда порождаем переход из I в τ (по символу τ).
- Если $\tau_1\tau_2 \in \mathrm{Next}(E')$, тогда порождаем переход из τ_1 в τ_2 по символу τ_2.
- Если $\tau \in \mathrm{Last}(E')$, тогда объявляем τ финальным.
- Стираем номера у символов на переходах. НКА, распознающий E, построен.

Построение 6. *Построим НКА, распознающий* $(a \mid (ab))^*b^+$.

- Линеаризуем: $E' = (a_1 \mid (a_2b_3))^*b_4^+$.
- Порождаем множества First, Last, Next:

$$\begin{aligned}&\mathrm{First}(E') = \{a_1, a_2, b_4\}\\ &\mathrm{Last}(E') = \{b_4\}\\ &\mathrm{Next}(E') = \{a_1a_1, a_1a_2, a_2b_3, b_3a_1, b_3a_2, a_1b_4, b_3b_4, b_4b_4\}\end{aligned}$$

- Строим конечный автомат:



Определение 13. *Множество* $a^{-1}U = \{w \mid aw \in U\}$ *называется производным Бзрозовски множества* U *относительно* a*. Если* $\varepsilon \in a^{-1}U$*, тогда* a *распознаётся выражением* U*.*

Λ_E положим равным $\{\varepsilon\}$, если $\varepsilon \in E$, и пустым множеством иначе.

Производные $\mathcal{RE}$

- $a^{-1}\varepsilon = \varnothing$, $a^{-1}\varnothing = \varnothing$;
- $a^{-1}a = \{\varepsilon\}$, $a^{-1}b = \varnothing$;
- $a^{-1}(\Phi \mid \Psi) = a^{-1}(\Phi) \cup a^{-1}(\Psi)$;
- $a^{-1}(\Phi\ \Psi) = a^{-1}(\Phi)\Psi \cup \Lambda_\Phi a^{-1}(\Psi)$;
- $a^{-1}(\Phi^*) = a^{-1}(\Phi)\Phi^*$.

С помощью последовательного взятия производных можно свести задачу $w \in \mathscr{L}(R)$ к задаче $\varepsilon \in w^{-1}R$. На этом построен ещё один способ преобразования $\mathcal{RE}$ к автомату.

Построение 7. *Рассмотрим всё то же выражение* $(a \mid (ab))^*b^+$*. Построим по нему автомат с помощью производных Брзозовски.*

Пример преобразования

- $a^{-1}(a \mid (ab))^*b^+ = (a^{-1}(a \mid (ab))^*)b^+ \cup (a^{-1}b^+)$, но второе очевидно пусто, поэтому $a^{-1}(a \mid (ab))^*b^+ = (\varepsilon \mid b)\ (a \mid (ab))^*b^+$;
- $b^{-1}(a \mid (ab))^*b^+ = (b^{-1}(a \mid (ab))^*)b^+ \cup (b^{-1}b^+)$, и здесь как раз пусто первое, поэтому производная равна b^*.
- $a^{-1}b^* = \varnothing$; $b^{-1}b^* = b^*$.
- $a^{-1}((\varepsilon \mid b)\ (a \mid (ab))^*b^+)$ вынуждает первую альтернативу в $(\varepsilon|b)$ и порождает само себя.
- $b^{-1}((\varepsilon \mid b)\ (a \mid (ab))^*b^+)$ порождает $(a \mid (ab))^*b^+ \mid b^*$.
- $a^{-1}((a \mid (ab))^*b^+ \mid b^*)$ порождает $(\varepsilon \mid b)\ (a \mid (ab))^*b^+$, $b^{-1}((a \mid (ab))^*b^+ \mid b^*)$ порождает b^*.
- Переходы замкнулись. Осталось собрать производные в состояния автомата.

Состояние	Производная
q_1	$(a \mid (ab))^*b^+$
q_2	$(\varepsilon \mid b)\ (a \mid (ab))^*b^+$
q_3	b^*
q_4	$(a \mid (ab))^*b^+ \mid b^*$

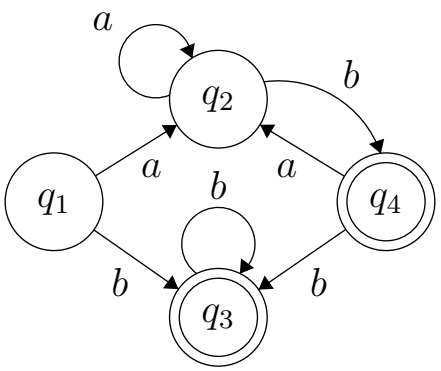


Рисунок 6 — К параграфу «Пример преобразования»

Неподвижная точка функции $f(x)$ — такое x, что $f(x) = x$.

Лемма 1. *Пусть $X = (AX) \mid B$, где X — неизвестное $\mathcal{RE}$, а A, B — известные, причём $\varepsilon \notin \mathscr{L}(A)$. Тогда $X = (A)^*B$.*

Рассмотрим систему уравнений:

$$\begin{array}{rcl} X_1 & = & (A_{11}X_1) \mid (A_{12}X_2) \mid \ldots \mid B_1 \\ X_2 & = & (A_{21}X_1) \mid (A_{22}X_2) \mid \ldots \mid B_2 \\ & \ldots & \\ X_n & = & (A_{n1}X_1) \mid (A_{n2}X_2) \mid \ldots \mid B_n \end{array}$$

Положим $\varepsilon \notin A_{ij}$. Будем последовательно выражать X_1 через X_2, …, X_n, X_2 через $X_3 \ldots X_n$ и т.д. Получим регулярное выражение для X_n.

Построение 8. *По каждому НКА можно построить $\mathcal{RE}$, распознающую тот же язык. Верно и обратное.*

От грамматики и НКА к $\mathcal{RE}$
Здесь считаем, что в НКА нет ε-переходов.

- Объявляем каждый нетерминал (или состояние НКА) переменной и строим для него уравнение:
  - По правилу $A \to aB$ (или для стрелки из A в B) добавляем альтернативу aB;
  - По правилу $A \to b$ (или для стрелки в финальное состояние) добавляем альтернативу без переменных.
  - Правило $S \to \varepsilon$ обрабатываем отдельно, не внося в уравнение: добавляем в язык альтернативу $(\mathcal{RE} \mid \varepsilon)$.
- Решаем систему относительно S.

Пример 7. *Построим $\mathcal{RE}$ по грамматике:*

$$\begin{array}{llll} S \to aT & S \to abS & & \\ T \to aT & T \to bT & T \to b & \end{array}$$

Строим по правилам грамматики систему: $S = (abS) \mid (aT)$
$T = ((a \mid b)T) \mid b$
Решаем второе уравнение:
$T = (a \mid b)^*b$
Подставляем в первое:
$S = (abS) \mid (a(a \mid b)^*b)$
Получаем ответ:
$S = (ab)^*a(a \mid b)^*b$

Представления регулярных языков. Критерий регулярности

Определение 14. *Недетерминированный конечный автомат (NFA) — это пятёрка $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$, где:*

- Q — *множество состояний;*
- Σ — *алфавит терминалов;*

- δ — *множество правил перехода вида* $\langle q_i, (a_i|\varepsilon), M_i\rangle$, *где* $q_i \in Q$, $a_i \in \Sigma$, $M_i \in 2^Q$;
- $q_0 \in Q$ — *начальное состояние;*
- $F \subseteq Q$ — *множество конечных состояний.*

.

Сокращаем: $\langle q_1, a, q_2\rangle \in \delta \Leftrightarrow \langle q_1, a, M\rangle \in \delta \ \&\ q_2 \in M$.

Недетерминированные КА

- $q \xrightarrow{\varepsilon} q' \Leftrightarrow (q = q') \vee \exists p_1, \ldots, p_k(\langle q, \varepsilon, p_1\rangle \in \delta \ \&\ \langle p_k, \varepsilon, q'\rangle \in \delta \ \&\ \forall i, 1 \le i < k \langle p_i, \varepsilon, p_{i+1}\rangle \in \delta)$.
- $q \xrightarrow{a} q' \Leftrightarrow \exists p, p'(q \xrightarrow{\varepsilon} p \ \&\ \langle p, a, p'\rangle \in \delta \ \&\ p' \xrightarrow{\varepsilon} q')$.
- $q \xrightarrow{a_1 \ldots a_k} q' \Leftrightarrow \exists p_1, \ldots, p_{k-1}(q \xrightarrow{a_1} p_1 \ \&\ p_{k-1} \xrightarrow{a_k} q' \ \&\ \forall i, 1 \le i < k-1(p_i \xrightarrow{a_{i+1}} p_{i+1}))$.

Определение 15. *Язык* $\mathscr{L}$, *распознаваемый НКА* $\mathscr{A}$ — *это множество слов* $\{w \mid \exists q \in F(q_0 \xrightarrow{w} q)\}$.

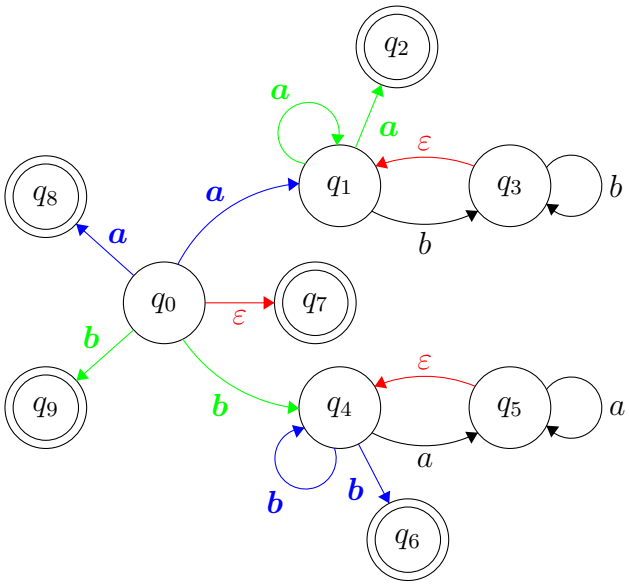


Рисунок 7 — Пример НКА

Определение 16. *Детерминированный конечный автомат (DFA) — это пятёрка* $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta\rangle$, *где:*

- Q — *множество состояний;*
- Σ — *алфавит терминалов;*
- δ — *множество правил перехода вида* $\langle q_i, a_i, q_j\rangle$, *где* $q_i, q_j \in Q$, $a_i \in \Sigma$, *причём* $\forall q_i, a_i \exists q_j(\langle q_i, a_i, q_j\rangle \in \delta \ \&\ \forall q_k(\langle q_i, a_i, q_k\rangle \in \delta \Rightarrow q_k = q_j))$;
- $q_0 \in Q$ — *начальное состояние;*
- $F \subseteq Q$ — *множество конечных состояний.*

ε-переходов нет $\Rightarrow q \xrightarrow{a} q' \Leftrightarrow \langle q, a, q'\rangle \in \delta$.

Определение 17. *Язык* $\mathscr{L}$, *распознаваемый* $\mathscr{A}$ — *это множество слов* $\{w \mid \exists q \in F(q_0 \xrightarrow{w} q)\}$.

Sink/trap state (состояние–ловушка)
«Ловушка» — не конечное состояние с переходами лишь в себя. Нужны для корректного задания DFA, но иногда по умолчанию не описываются.

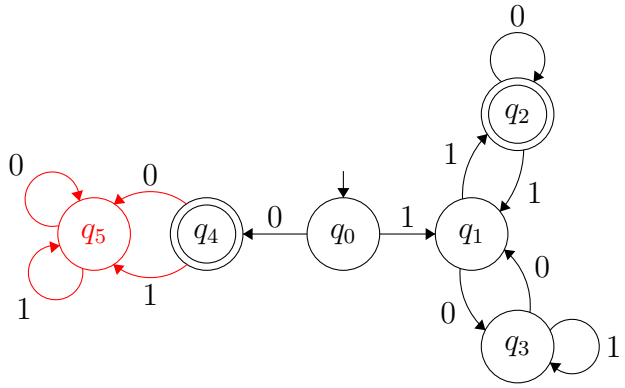
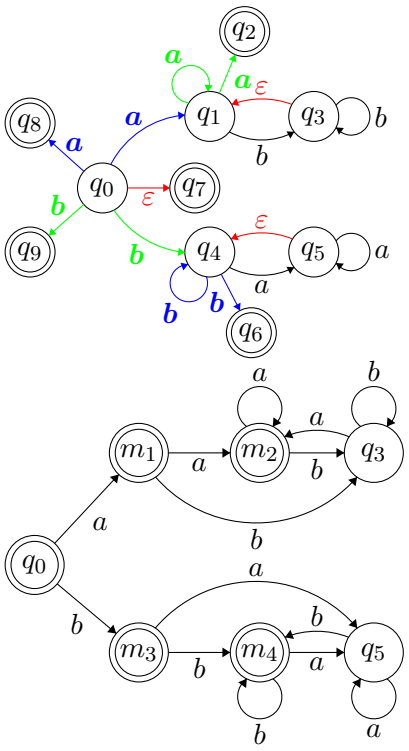


Рисунок 8 — К параграфу «Sink/trap state (состояние–ловушка)»

Определение 18. *Состояния DFA $D(\mathscr{A})$ — это состояния $m_i \in 2^Q$, где Q — состояния NFA $\mathscr{A}$.*

- $m_0 = \{q_i \mid q_0 \xrightarrow{\varepsilon} q_i\}$;
- $m_i \in F_D \Leftrightarrow \exists q_i, q_j \{q_i \in m_i \;\&\; q_j \in F(\mathscr{A}) \;\&\; q_i \xrightarrow{\varepsilon} q_j\}$;
- $\langle m, a, m' \rangle \in \delta_D \Leftrightarrow m' = \{q_i \mid \exists q_j \in m (q_j \xrightarrow{a} q_i)\}$.



- $\{q_0\} \xrightarrow{a} \{q_1, q_8\}$, $\{q_0\} \xrightarrow{b} \{q_4, q_9\}$;
- $\{q_1, q_8\} \xrightarrow{a} \{q_1, q_2\}$, $\{q_1, q_8\} \xrightarrow{b} \{q_3\}$; $\{q_1, q_8\} \sim m_1$.
- $\{q_1, q_2\} \xrightarrow{a} \{q_1, q_2\}$, $\{q_1, q_2\} \xrightarrow{b} \{q_3\}$; $\{q_1, q_2\} \sim m_2$.
- $\{q_3\} \xrightarrow{a} \{q_1, q_2\}$, $\{q_3\} \xrightarrow{b} \{q_3\}$;
- $\{q_4, q_9\} \xrightarrow{b} \{q_4, q_6\}$, $\{q_4, q_9\} \xrightarrow{a} \{q_5\}$; $\{q_4, q_9\} \sim m_3$;
- $\{q_4, q_6\} \xrightarrow{b} \{q_4, q_6\}$, $\{q_4, q_6\} \xrightarrow{a} \{q_5\}$; $\{q_4, q_6\} \sim m_4$.
- $\{q_5\} \xrightarrow{b} \{q_4, q_6\}$, $\{q_5\} \xrightarrow{a} \{q_5\}$.

Рисунок 9 — Пример детерминизации

Гомоморфизм над свободной полугруппой (множеством слов) полностью определяется значениями на буквах, поскольку по определению $h(a_1 \circ a_2 \circ \cdots \circ a_n) = h(a_1) \circ h(a_2) \circ \cdots \circ h(a_n)$.
Здесь $\circ$ —конкатенация.

Замыкания регулярных языков

Пусть $\mathscr{L}$ — регулярный язык над Σ. Тогда регулярны:

- язык $\Sigma^* \setminus \mathscr{L}$;
- для любого гомоморфизма h язык $\{h(w) \mid w \in \mathscr{L}\}$;
- для любого гомоморфизма h язык $\{w \mid h(w) \in \mathscr{L}\}$.

Рассмотрим DFA $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta\rangle$, распознающий $\mathscr{L}$. Построим $\mathscr{A}' = \langle Q, \Sigma, q_0, Q \setminus F, \delta\rangle$. Тогда $w \notin \mathscr{L} \Leftrightarrow w \in \mathscr{L}(\mathscr{A}')$. Рассмотрим регулярное выражение R такое, что $\mathscr{L}(R) = \mathscr{L}$. Заменим в нём все $a_i \in \Sigma$ на $h(a_i)$. Полученное таким образом выражение R' также регулярно, причём $\mathscr{L}(R') = h(\mathscr{L})$. Рассмотрим DFA $\mathscr{A} = \langle Q, \Sigma, q_0, F, \delta\rangle$, распознающий $\mathscr{L}$. Построим $\mathscr{A}' = \langle Q, \Sigma, q_0, F, \delta'\rangle$ такой, что $\langle q_i, a, q_j\rangle \in \delta' \Leftrightarrow q_i \xrightarrow{h(a)} q_j$ в исходном автомате $\mathscr{A}$.

Примеры

Рассмотрим язык $\mathscr{L}' = \{a^n b^m \mid n \neq m\}$.

Предположим, $\mathscr{L}'$ регулярен. Тогда $a^*b^* \setminus \mathscr{L}' = \{a^n b^n\}$ также регулярен, а мы знаем, что это не так. $\perp$

Рассмотрим язык $\mathscr{L}^f = \{(abaabb)^n b^n\}$.

Попытка доказать его нерегулярность леммой о накачке породит перебор по накачиваемым строкам $(abaabb)^+$, $(abaabb)^*a$, $(abaabb)^*ab$,$(abaabb)^*aba$, $(abaabb)^*abaa$, ... Рассмотрим гомоморфизм $h(a) = abaabb$, $h(b) = b$. $h^{-1}(\mathscr{L}^f) = \{a^n b^n\}$, который был бы регулярен, если бы L^f был регулярен. $\perp$

Пусть дан DFA $\mathscr{A}$. Положим $w_1 \equiv_{\mathscr{A}} w_2 \Leftrightarrow \exists q_i (q_0 \xrightarrow{w_1} q_i \;\&\; q_0 \xrightarrow{w_2} q_i)$.

Если $w_1 \equiv_{\mathscr{A}} w_2$, тогда $\forall z (w_1 z \in \mathscr{L}(\mathscr{A}) \Leftrightarrow w_2 z \in \mathscr{L}(\mathscr{A}))$.

Рассмотрим более общее отношение. Положим $w_1 \equiv_{\mathscr{L}} w_2 \Leftrightarrow \forall z (w_1 z \in \mathscr{L} \Leftrightarrow w_2 z \in \mathscr{L})$. Это отношение разбивает $\mathscr{L}$ на классы эквивалентности.

Теорема 3. *Язык $\mathscr{L}$ регулярен тогда и только тогда, когда множество его классов эквивалентности по $\equiv_{\mathscr{L}}$ конечно.*

Теорема 4. *Язык $\mathscr{L}$ регулярен тогда и только тогда, когда множество классов эквивалентности по $\equiv_{\mathscr{L}}$ конечно.*

$\Rightarrow$: Пусть $\mathscr{L}$ регулярен. Тогда он порождается некоторым DFA $\mathscr{A}$ с конечным числом состояний N. Значит, множество $\{q_i \mid q_0 \xrightarrow{w} q_i\}$ конечно, а для каждых двух w_1, w_2 таких, что $q_0 \xrightarrow{w_1} q_i$ и $q_0 \xrightarrow{w_2} q_i$, выполняется $w_1 \equiv_{\mathscr{L}} w_2$.

$\Leftarrow$: Пусть все слова в Σ^* принадлежат N классам эквивалентности $A_1, \ldots, A_n$ по $\equiv_{\mathscr{L}}$. Построим по ним DFA $\mathscr{A}$, распознающий $\mathscr{L}$. Классы A_i объявим состояниями.

Критерий регулярности языка

- Начальным состоянием объявим класс эквивалентности A_0 такой, что $\varepsilon \in A_0$.
- Конечными объявим такие A_j, что $\forall w \in A_j (w \in \mathscr{L})$.
- Если $w \in A_i$, $w\ a_k \in A_j$, тогда добавляем в δ правило $\langle A_i, a_k, A_j\rangle$. $\forall w_1, w_2 \in A_i$, $w_1 a_k$ и $w_2 a_k$ всегда принадлежат одному и тому же A_j.

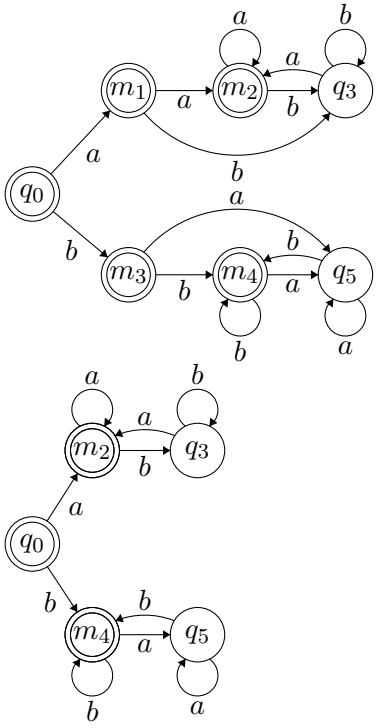
Построение 9. *1. Построим таблицу всех двухэлементных множеств $\{q_i, q_j\}$, $q_i, q_j \in Q$.*

2. Пометим все множества $\{q_i, q_j\}$ такие, что одно из q_i, q_j из F, а второе нет.

3. Пометим все множества $\{q_i, q_j\}$ такие, что $\exists a (q_i \xrightarrow{a} q_1' \;\&\; q_j \xrightarrow{a} q_2' \;\&\; \{q_1', q_2'\}$ — помеченная пара).

4. Продолжаем шаг 3, пока не будет появляться новых помеченных пар.

Пары, оставшиеся непомеченными, можно объединить.



m_1	✓					
m_2	✓					
q_3	✓	✓	✓			
m_3	✓	✓	✓	✓		
m_4	✓	✓	✓	✓		
q_5	✓	✓	✓	✓	✓	✓
	q_0	m_1	m_2	q_3	m_3	m_4

Рисунок 10 — Пример минимизации

$$q_0 \xrightarrow{a} m_1, m_1 \xrightarrow{a} m_2 \quad q_0 \xrightarrow{b} m_3, m_1 \xrightarrow{b} q_3 \quad \{m_1, m_2\} \xrightarrow{a} m_2 \quad \{m_1, m_2\} \xrightarrow{b} q_3$$
$$q_0 \xrightarrow{a} m_1, m_2 \xrightarrow{a} m_2 \quad q_0 \xrightarrow{b} m_3, m_2 \xrightarrow{b} q_3$$

$$q_0 \xrightarrow{a} m_1, m_3 \xrightarrow{a} q_5 \quad q_0 \xrightarrow{a} m_1, m_4 \xrightarrow{a} q_5 \quad m_1 \xrightarrow{a} m_2, m_3 \xrightarrow{a} q_5$$
$$m_2 \xrightarrow{a} m_2, m_3 \xrightarrow{a} q_5 \quad m_1 \xrightarrow{a} m_2, m_4 \xrightarrow{a} q_5 \quad m_2 \xrightarrow{a} m_2, m_4 \xrightarrow{a} q_5$$

$$\{m_3, m_4\} \xrightarrow{a} q_5 \qquad \{m_3, m_4\} \xrightarrow{b} m_4$$
$$q_3 \xrightarrow{a} m_2, q_5 \xrightarrow{a} m_4$$

Можно объединить состояния m_1 и m_2 и состояния m_3 и m_4.

Меньше чем пятью состояниями не обойтись. Рассмотрим слова ε, a, b, ab, ba. Каждые два из них различаются по $\equiv_{\mathscr{L}}$ при выборе одного из трёх z: ε, a или b.

Бисимуляция

Скажем, что состояния s_1, s_2 системы переходов $\mathscr{A}$ находятся в отношении бисимуляции ($s_1 \sim s_2$), если выполняются условия:

- $\forall t_1, a(s_1 \xrightarrow{a} t_1 \Rightarrow \exists t_2(s_2 \xrightarrow{a} t_2 \;\&\; t_1 \sim t_2))$;
- $\forall t_2, a(s_2 \xrightarrow{a} t_2 \Rightarrow \exists t_1(s_1 \xrightarrow{a} t_1 \;\&\; t_1 \sim t_2))$.

Бисимуляция — более сильное свойство, чем эквивалентность!

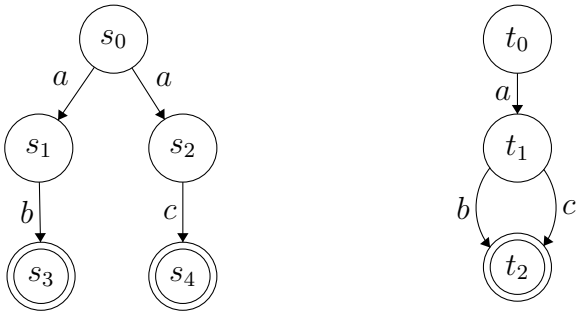


Рисунок 11 — К параграфу «Бисимуляция»

Определение 19. *Пусть $w^{-1}U$ — это производная U по w, т.е. $\{v \mid wv \in U\}$. Тогда выполнено $x \equiv_U y \Leftrightarrow x^{-1}U = y^{-1}U$.*

Связь M.– N. и производных

- Количество производных (как языков) регулярного языка конечно.
- Конструкция Брзозовки порождает минимальный DFA.

Но проблема с правилами переписывания (ACI):

- $(w_1 \mid w_2) \mid w_3 = w_1 \mid (w_2 \mid w_3)$
- $w_1 \mid w_2 = w_2 \mid w_1$
- $w \mid w = w$

Применение теоремы M.– N.
Дан язык $\mathscr{L}$. Показать, что он не регулярен, пользуясь теоремой Майхилла–Нероуда.

Стандартный подход

1. Подобрать бесконечную последовательность префиксов $w_1,\ldots, w_n, \ldots$.
2. Подобрать бесконечную последовательность суффиксов $z_1,\ldots, z_n, \ldots$, такую, что $w_i{+}{+}\, z_i \in \mathscr{L}$.
3. Доказать, что в таблице конкатенаций все строки различны (значит, $\forall i, j\, \exists k(w_i z_k \in \mathscr{L}\ \&\ w_j z_k \notin \mathscr{L})$).

Диагональная конструкция (условие $w_i + + z_i \in \mathscr{L}$) — одна из многих возможных, обычно она довольно удобна.

Диагональная конструкция
Рассмотрим язык $L = \{a^n b^n\}$. Положим $w_i = a^i$, $z_i = b^i$. Тогда таблица конкатенаций w_i, z_j будет выглядеть следующим образом. Здесь $+$ — это то же, что «$\in \mathscr{L}$», $-$ читаем как «$\notin \mathscr{L}$».

	$z_1 = b$	$z_2 = b^2$	$z_3 = b^3$	$\ldots$	$z_n = b^n$	$\ldots$
$w_1 = a$	$+$	$-$	$-$		$-$	
$w_2 = a^2$	$-$	$+$	$-$		$-$	
$w_3 = a^3$	$-$	$-$	$+$		$-$	
$\ldots$			$\ldots$			
$w^n = a^n$	$-$	$-$	$-$		$+$	
$\ldots$						

Так же можно обосновывать минимальность DFA. Рассмотрим минимальный автомат из примера выше. Его язык — слова в $\{a, b\}^*$, начинающиеся и заканчивающиеся одной и той же буквой. Построим таблицу классов эквивалентности по $w_i \in \{\varepsilon, a, b, ab, ba\}$.

	ε	a	b
ε	$+$	$+$	$+$
a	$+$	$+$	$-$
b	$+$	$-$	$+$
ab	$-$	$+$	$-$
ba	$-$	$-$	$+$

В этой таблице все строчки различны, значит, выбранные w_i действительно лежат в различных классах эквивалентности, и DFA, распознающий язык $\mathscr{L}$, не может иметь меньше пяти состояний.

При доказательстве минимальности DFA достаточно подобрать $[log_2 n] + 1$ различающих суффиксов z_i, где n — число состояний автомата.

О порождении новых алгоритмов

Пусть $\mathscr{A}$ — NFA. Тогда $det(reverse(det(reverse(\mathscr{A}))))$ — минимальный DFA, эквивалентный $\mathscr{A}$.

Многие алгоритмы для порождения малых (не минимальных) NFA являются комбинациями нескольких базовых операций.

- Обращение автомата
- Детерминизация
- Удаление ε-правил
- Минимизация
- Разметка

Автомат Томпсона

- Единственное начальное состояние
- Единственное конечное состояние
- Не больше двух переходов из каждого состояния

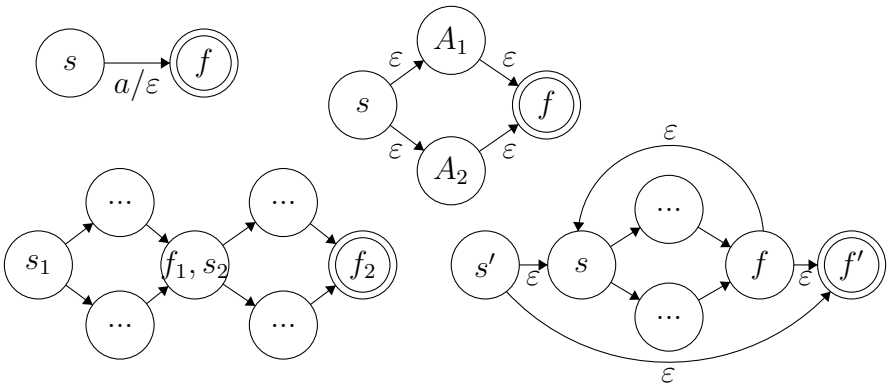


Рисунок 12 — К параграфу «Автомат Томпсона»

Несколько конструкций

- Автомат Глушкова: $rmeps(Th(R))$;
- Автомат Антимирова: $rmeps(deannote(minimize(rmeps(annote\_eps(Th(R))))))$;
- Автомат Илия–Ю: $deannote(minimize(rmeps(annote(Th(R)))))$.

Другие регулярные модели. Синтаксический моноид

Леволинейные грамматики
Класс грамматик, симметричный праволинейным:

- виды правил — $A_i \to a_j$, $A_i \to A_k a_j$ и $S \to \varepsilon$, если S не встречается в правых частях правил;
- описывает тот же самый класс языков, что и праволинейные грамматики.

Преобразование в леволинейную форму легко сделать по обратным проходам из финального в начальное состояние в НКА, соответствующем грамматике.

$$S \to aA \,|\, bB \quad A \to aS \,|\, a \quad B \to bS \,|\, b$$

Строим недетерминированный КА по грамматике. Здесь F — финальное состояние, куда добавляются переходы $\langle A_i, a_j \rangle \to F$, соответствующие правилам $A_i \to a_j$ грамматики.

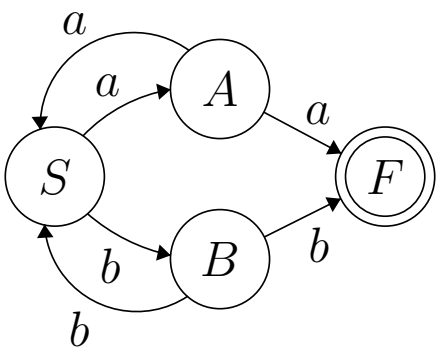


Рисунок 13 — К параграфу «Леволинейные грамматики»

Теперь обращаем стрелки и меняем начальное и финальное состояния местами.

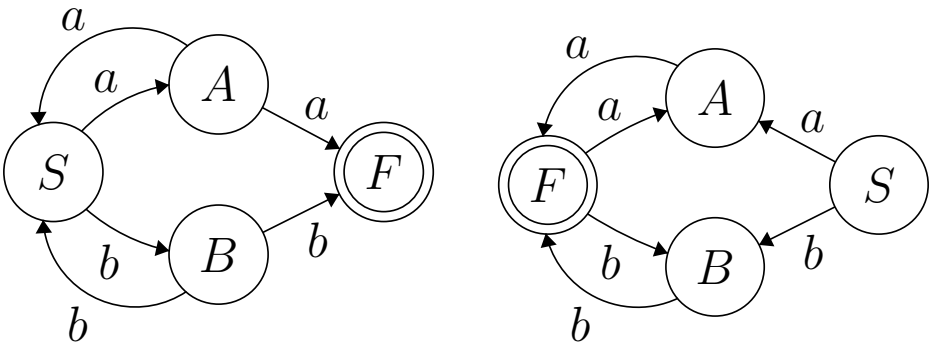


Рисунок 14 — К параграфу «Леволинейные грамматики»

По полученному автомату строим леволинейную грамматику:

$$S \to Aa \,|\, Bb \quad A \to Fa \,|\, a \quad B \to Fb \,|\, b \quad F \to Aa \,|\, Bb$$

Формальный алгоритм приведения к леволинейности

1. Добавляем в грамматику новый стартовый символ S' и для всех правил вида $A_i \to a_j$ — правила $S' \to A_i a_j$.
2. Правила вида $A_i \to a_j A_k$ преобразуем в $A_k \to A_i a_j$.
3. По правилам вида $S \to a_j A_k$ дополнительно порождаем правила $A_k \to a_j$.
4. По правилам вида $S \to a_j$ и $S \to \varepsilon$ порождаем правила $S' \to a_j$ и $S' \to \varepsilon$ соответственно.

Если в исходной грамматике S не встречался в правых частях правил, тогда этот алгоритм породит непродуктивные правила $A_k \to S a_j$. Поэтому шаг 2 для правил вида $S \to a_j A_k$ и шаг 1 для $S \to a_j$ в этом случае делать не надо.

Предложение 7. *Если $\mathscr{L}$ регулярен, то и $\mathscr{L}^R$ регулярен.*

Очевидно для НКА (обращаем стрелки в НКА без перемены стартовых и финальных состояний), также очевидно для regex (почему?)

А что с ДКА?

Минимизация по Брзозовски
$det(reverse(det(reverse(\mathscr{A}))))$ является минимальным ДКА для любого НКА $\mathscr{A}$.

Реверсирование принципиально меняет структуру минимального автомата. Причина — асимметричность определения классов эквивалентности:

$$u \equiv_{\mathscr{L}} w \Leftrightarrow \forall x (u\,x \in \mathscr{L} \Leftrightarrow v\,x \in \mathscr{L})$$

Построение 10. *Имея регулярную грамматику с N нетерминалами, по ней можно построить ДКА (самое большее) с $O(2^N)$ состояниями. Эта оценка является точной.*

Цена детерминизма
Рассмотрим грамматику G:

$$\begin{array}{llll} S \to aS & S \to bS & S \to bA_1 & \\ A_1 \to aA_2 & A_1 \to bA_2 & A_2 \to aA_3 & A_2 \to bA_3 \\ \dots & & & \\ A_{n-1} \to aA_n & A_{n-1} \to bA_n & A_n \to a & A_n \to b \end{array}$$

В отличие от теоремы Майхилла–Нероуда, лемма о накачке использует свойства НКА, а не ДКА. А именно, если константа накачки языка не может быть меньше k, то в НКА, распознающем этот язык, не меньше k состояний.

Ещё раз о лемме о накачке
Рассмотрим язык $L = \{w_1 b w_2 \,|\, |w_2| = 3\}$. Мы знаем, что распознающий его ДКА имеет минимум 16 состояний. Накачку $w \in L$, такую что $w = xyz$, $xy^n z \in L$, найти очень просто для всякого слова длины ≥ 5: достаточно взять первую букву этого слова в качестве y, а x принять пустым.

Теперь пусть длина накачки p меньше 5. Рассмотрим слово $baaa \in L$. Любая накачка только букв a (нулевая и нет) выводит слово из языка, и нулевая накачка подслова, содержащего букву b, также выводит из языка L. Поэтому НКА, распознающий L, не может иметь меньше 5 состояний.

Посмотрим на правила перехода: $\langle q_1, a\rangle \to q_2$. Если частично специализировать их по элементам из Σ, то получится функция $F_a : Q \to Q$ (Q — множество состояний автомата). Такие же функции можно определить для слов по композиции.

Предложение 8. *Каждый автомат $\mathscr{A}$ определяет моноид $\mathcal{M} = \{w \mid w \in \Sigma^+\}$ такой, что $w_i = w_j \Leftrightarrow F_{w_i} = F_{w_j}$. Классы эквивалентности слов в $\mathcal{M}$ соответствуют функциям F_{w_i}.*

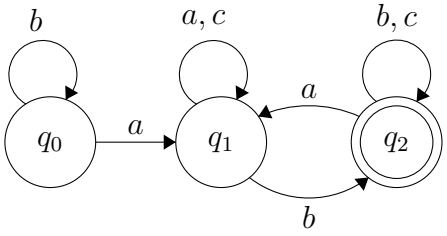


Рисунок 15 — Пример построения Т.М.

Определим соответствие между буквами и множествами переходов по ним и будем расширять этот список новыми словами в лексикографическом порядке.

$$a := \{(0,1),(1,1),(2,1)\} \quad b := \{(0,0),(1,2),(2,2)\} \quad c := \{(1,1),(2,2)\}$$
$$ab := \{(0,2),(1,2),(2,2)\} \quad bc := \{(1,2),(2,2)\} \quad ca := \{(1,1),(2,1)\}$$

$aa \to a \qquad ac \to a \qquad ba \to a$
$bb \to b \qquad cc \to c \qquad cb \to bc$
$abc \to ab \qquad bca \to ca \qquad cab \to bc$

Предложение 9. *Положим $u \sim_{\mathscr{L}} v \Leftrightarrow \forall x, y(x\,u\,y \in \mathscr{L} \Leftrightarrow x\,v\,y \in \mathscr{L})$.*
Синтаксический моноид $\mathcal{M}(\mathscr{L})$: $\{w \mid w_i = w_j \Leftrightarrow w_i \sim_{\mathscr{L}} w_j\}$.

Синтаксический моноид

Синтаксический моноид регулярного языка $\mathscr{L}$ совпадает с трансформационным моноидом минимального ДКА, его распознающего.

Одному классу эквивалентности синтаксического моноида может соответствовать несколько классов эквивалентности трансформационного, но не наоборот.

Определение 20. *ДКА $\mathscr{A}$ называется синхронизирующимся, если $\exists w, q_s \forall q_i (q_i \xrightarrow{w} q_s)$.*

Предложение 10. *ДКА $\mathscr{A}$ синхронизирующийся $\Leftrightarrow \forall q, q' \exists w, q_x (q \xrightarrow{w} q_x \;\&\; q' \xrightarrow{w} q_x)$.*

Построение 11. *Рассмотрим слово w_1, синхронизирующее q_1 и q_2. Если w_1 синхронизирует все состояния, доказывать нечего. Иначе построим множество $Q_1 = \{q \mid q_i \xrightarrow{w_1} q\}$. По построению, $Q_1 \subset \{q_1, \ldots, q_n\}$. Выберем в нём два первых состояния, q_i, q_j, и слово w_2, синхронизирующее их. Построим множество $Q_2 = \{q \mid q_i \in Q_1 \;\&\; q_i \xrightarrow{w_2} q\}$. По построению, $Q_2 \subset Q_1$. Продолжив так не более чем $n-1$ раз, построим синхронизирующее слово.*

Предложение 11. *ДКА синхронизируется $\Leftrightarrow$ классы эквивалентности его трансформационного моноида содержат «константу», т.е. класс, переводящий все состояния в одно.*

Задача Эшби о привидениях

Дорогой друг! Недавно я купил старый дом, в котором обитают два призрака: Певун и Хохотун. Я установил, что их поведение подчиняется определенным законам, и что я могу воздействовать на них, играя на органе или сжигая ладан. В течение каждой минуты каждый из призраков либо шумит, либо молчит. Поведение же их в каждую минуту зависит только от минуты до этого, и эта зависимость такова.

Певун всегда ведет себя так же, как и в предыдущую минуту (звучит или шумит), если только в эту предыдущую минуту не было игры на органе при молчании Хохотуна. В последнем случае Певун меняет свое поведение на противоположное. Что касается Хохотуна, то, если в предыдущую минуту горел ладан, он будет вести себя так же, как Певун минутой раньше. Если, однако, ладан не горел, Хохотун будет вести себя противоположно Певуну в предыдущую минуту. Что мне делать, чтобы установить и поддерживать тишину в доме?

- Если не играли на органе или Хохотун шумел, Певун не меняет поведение, иначе меняет.
- Если горел ладан, Хохотун делает то же, что делал Певун, иначе — противоположное.

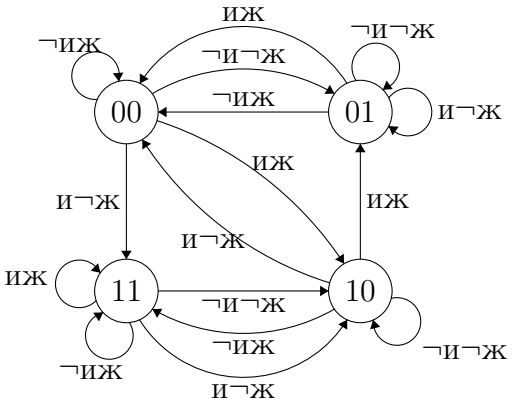


Рисунок 16 — К параграфу «Задача Эшби о привидениях»

Синхронизирующее к состоянию 00 слово: ¬и¬ж, и¬ж, ¬иж.

Определение 21. *Двоичное префиксное кодирование — это гомоморфизм* $h : \Sigma^+ \to \{0,1\}^+$ *такой, что* $\forall a, b \in \Sigma\, \forall w \in \{0,1\}^*(h(a) \neq h(b)w)$.

Префиксное кодирование

Рассмотрим префиксный код из 9-буквенного алфавита: $\mathcal{C} = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$.

Автомат–декодер для $\mathcal{C}$ (возвращается в ε-состояние, дочитав очередной код):

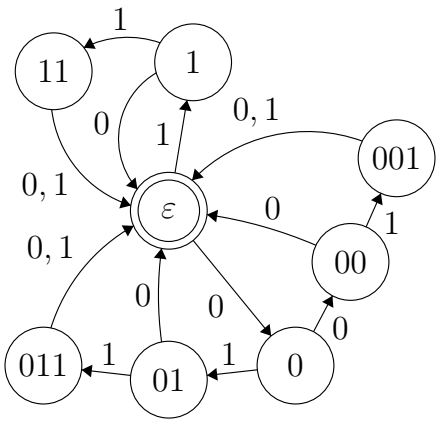


Рисунок 17 — К параграфу «Префиксное кодирование»

Префиксный код максимален, если к множеству кодирующих слов нельзя добавить ни одно слово без нарушения префикс-свойства (т.е. запрета слов из множества быть префиксами друг друга).

Максимальный префиксный двоичный код $\mathcal{C}$ называют синхронизированным, если $\exists z \in \{0,1\}^+$, такое что $\forall y \in \{0,1\}^+$ слово yz можно представить как конкатенацию слов из $\mathcal{C}$.

Если код $\mathcal{C}$ синхронизирован, тогда ошибки в передаче закодированного слова будут исправляться сами при передаче достаточно длинной закодированной последовательности.

Предложение 12. *Максимальный префиксный код синхронизирован $\Leftrightarrow$ его декодер — синхронизирующийся ДКА.*

Определение 22. *Дана SRS $\mathcal{S}$ с правилами переписывания двух видов:*

$$a_i \to b_1 \dots b_n \qquad a_i \to \varepsilon$$

Разрешим применять правила только к первым буквам слова. Пусть дана пара $\langle \mathcal{S}, w_0 \rangle$, где w_0 — слово в алфавите Σ. Эта пара определяет алфавитную префиксную грамматику.

Алфавитные префиксные грамматики

Язык $L\langle \mathcal{S}, w_0 \rangle$ регулярен.

Скажем, что $a \twoheadrightarrow \varepsilon$ (a коллапсирует), если либо $a \to \varepsilon \in \mathcal{S}$, либо $\exists b_1, \dots, b_n (\forall b_i (b_i \twoheadrightarrow \varepsilon)\ \&\ a \to b_1 \dots b_n \in \mathcal{S})$.

По APG $\langle \mathcal{S}, s_1 \dots s_n \rangle$ породим праволинейную грамматику G. Каждому символу алфавита a_i сопоставим A_i — нетерминал G.

1. Пусть $a \to b_1 \dots b_n$ и $\exists b_i (\neg(b_i \twoheadrightarrow \varepsilon)\ \&\ \forall j (j < i \Rightarrow b_j \twoheadrightarrow \varepsilon))$. Тогда добавим в G правила $A \to B_1 b_2 \dots b_n$, $A \to B_2 b_3 \dots b_n$,..., $A \to B_i b_{i+1} \dots b_n$, $A \to a$.
2. Если такого b_i нет, добавляем в G все правила вида $A \to B_1 b_2 \dots b_n$, ..., $A \to B_{n-1} b_n$, $A \to B_n$, $A \to a$.
3. Вводим стартовый нетерминал S и для него добавляем развёртку в исходное слово $s_1 \dots s_m$ по правилам выше.
4. Если все s_i коллапсируют, тогда добавляем в G правило $S \to \varepsilon$.

Остается сделать развертку правил вида $A \to B_n$, либо перейти от G к НКА с ε-переходами.

Если вместо правил $a_i \to b_1 \dots b_n$ к префиксам слов можно применять любые правила вида $a_1 \dots a_m \to b_1 \dots b_n$, такая грамматика называется (просто) префиксной. Для простоты предполагаем, что начальное слово также может быть не единственным.

Неалфавитные грамматики

Языки префиксных грамматик регулярны.

Доказательство использует ту же идею, что в случае АПГ: множество минимальных укорачивающихся комбинаций правил переписывания конечно.

В данном алгоритме рассматривается минимальный ДКА для языка.

От ДКА к префиксной грамматике

- Ведущими словами для нетерминалов (состояний) q_i объявим классы эквивалентности w_i такие, что $q_0 \xrightarrow{w_i} q_i$.
- Для всех стрелок, входящих в q_i из q_k и помеченных буквами a_j, построим правила переписывания: $w_i \to w_k a_j$.
- Начальными словами объявим слова из классов эквивалентности, лежащих в языке автомата.

Построим префиксную грамматику для языка уже знакомого нам автомата:

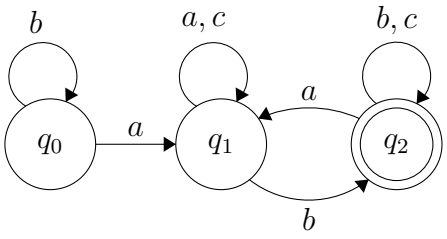


Рисунок 18 — К параграфу «От ДКА к префиксной грамматике»

Для q_0 ведущим словом будет b, для q_1 — a, для q_2 ведущее ab (оно же стартовое слово).

Правила префиксной грамматики:

Стартовое слово: ab. Правила переписывания:

$$
\begin{array}{ll}
b \to bb & \text{(в } q_0 \text{ входит лишь одна стрелка)} \\
a \to aa \quad a \to ac & \text{(стрелки из } q_1 \text{ в себя)} \\
a \to ba & \text{(стрелка из } q_0 \text{ в } q_1\text{)} \\
ab \to ab & \text{(стрелка из } q_1 \text{ в } q_2\text{)} \\
ab \to abc \quad ab \to abb & \text{(стрелки из } q_2 \text{ в себя)}
\end{array}
$$

Результат похож на обращенные правила трансформационного моноида, но учитывает префиксность: нет смысла переписывать $c \to cc$, если c может встретиться только после буквы a, либо после префикса ab.

Рассмотрим стек с вершиной $\bullet_n$:

$\bullet_n \leftarrow f_{n+1}(\dots)$, $\bullet_{n-1} \leftarrow f_n(\bullet_n \dots) \dots$, $\bullet_0 \leftarrow f_1(\bullet_1 \dots)$

Опишем его состояние перечислением имён функций в порядке их вхождения: $f_{n+1} f_n \dots f_1$.

Шаги вычислений над такими состояниями стека описываются как применения правил в APG.

«Подозрительное» поведение — такое, при котором вершина стека повторяется, выбрасывая промежуточные вычисления.

Отношение Турчина

Пусть на пути развертки программы имеются два состояния стеков: $c_1 : \Phi\Theta_0$, $c_2 : \Phi\Psi\Theta_0$, такие что Θ_0 неизменна на всём отрезке пути от c_1 до c_2. Тогда скажем, что $c_1 \preceq c_2$ (связаны отношением Турчина).

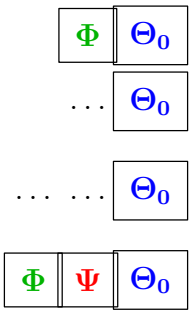


Рисунок 19 — Поиск бесконечных циклов

Если вершина Φ действительно входит в бесконечный цикл, порождая всё новые состояния вида $\Phi\Psi^n\Theta_0$, тогда правдоподобно, что $c_1 \preceq c_2$. Однако может случиться, что $c_1 \preceq c_2$ и на развертке завершающегося вычисления (ложное срабатывание).

Вариант для CBV

На любом бесконечном пути вычислений имеются два состояния стека, такие что $c_1 \preceq c_2$.

Теорема Турчина гарантирует, что существование $\preceq$-пар — необходимое условие бесконечного (зацикливающегося) вычисления. Поэтому $\preceq$ может использоваться для приблизительного анализа завершаемости программ (наряду с другими условиями).

Определение 23. *Пусть дано множество одноместных операций $\mathcal{V}_x = \mathcal{O}_x \cup \mathcal{P}_x$, задаваемое для участника x, причём для некоторых $p_1, p_2 \in \mathcal{V}_x$ выполняются тождества $p_1 \circ p_2 = id$, и для всех $p_1, p_2, p_3((p_1 \circ p_2) \circ p_3 = p_1 \circ (p_2 \circ p_3))$. Пинг-понг протокол для двух участников — это конечная последовательность инструкций $[p_1 \dots p_n, [x, y]]$, $p_i \in \mathcal{V}_x \cup \mathcal{O}_y$.*

$\mathcal{O}_x$ — публичные операции; $\mathcal{P}_x$ — приватные операции.

Д. Долев & А. Яо — первая формальная модель угрозы и первое формальное понятие криптографического протокола (1983).

Злоумышленник по Долеву–Яо:

- Может перехватывать, пересылать и изменять любое сообщение в сети;
- Может играть роль любого пользователя (маскарад);
- Может убедить пользователей начать любой дозволенный протоколом сеанс передачи сообщений.
- Не может совершать битовые операции над сообщениями;
- Не может угадать свойства секретных операций.

Рисунок 20 — Модель угрозы Долева–Яо

Легальные пользователи — **A**, **B**. Злоумышленник — **Z** (одного всегда достаточно).

Изначальное сообщение — M (обычно засекреченное).

Σ_x — словарь операторов x. E_x — зашифровка открытым ключом x, D_x — расшифровка E_x, a_x — приписывание к сообщению имени x, d_x — удаление префикса сообщения, совпадающего с именем x.

Протокол — набор α_i (слов протокола) и указаний, кто посылает α_i. Атака — последовательность подстановок в α_i, порождающая пустое слово (т.е. демаскирующая сообщение M).

Пример протокола

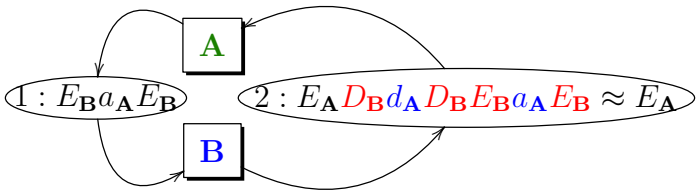
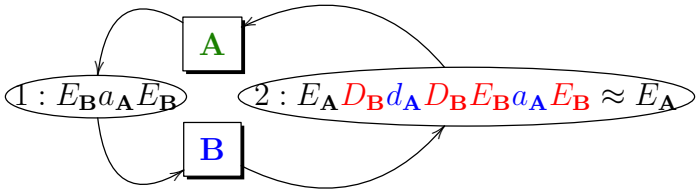
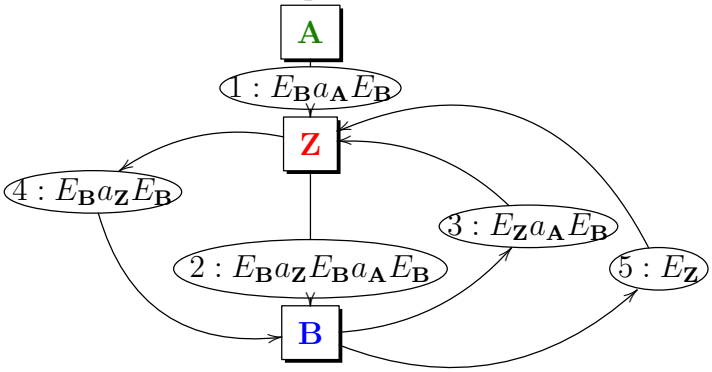


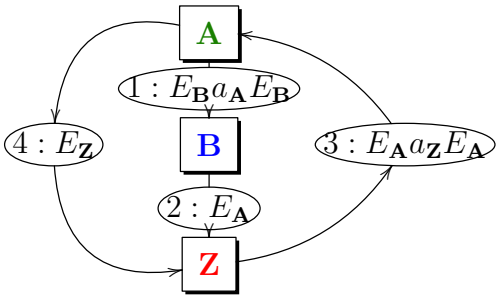
Рисунок 21 — Протокол для двух участников



Первая атака



Вторая атака

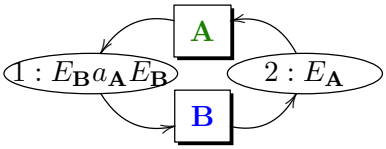


Автоматная модель $\mathscr{A}(P)$

- Строим все возможные подстановки в протокол P пар участников (включая злоумышленника);
- Строим начальное состояние 0 и конечное состояние 1, между ними — путь, соответствующий обращению первого слова протокола с двумя легальными участниками **A**, **B** (чтобы было что атаковать);
- Строим пути из 0 в 0, соответствующие реверсам (обращенным) словам-подстановкам в протокол P;
- Строим пути из 0 в 0, соответствующие всем возможным индивидуальным действиям злоумышленника **Z** — т.е. элементам $\mathcal{O}_{\mathbf{A}}$, $\mathcal{O}_{\mathbf{B}}$, $\mathcal{O}_{\mathbf{Z}}$ и $\mathcal{P}_{\mathbf{Z}}$.

Протокол P ненадёжен в модели угрозы Долева–Яо тогда и только тогда, когда $\varepsilon \in L(\mathscr{A}(P))$.

Протокол



Случаи
$\mathbf{P}_{\text{Double}}[\mathbf{A}, \mathbf{B}]$
$\mathbf{P}_{\text{Double}}[\mathbf{B}, \mathbf{A}]$
$\mathbf{P}_{\text{Double}}[\mathbf{A}, \mathbf{Z}]$
$\mathbf{P}_{\text{Double}}[\mathbf{Z}, \mathbf{B}]$

Автоматная модель

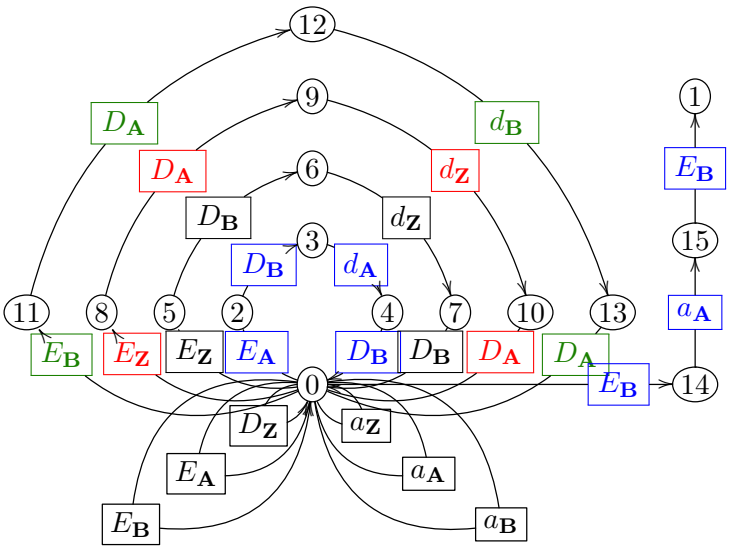


Рисунок 22

Контекстно-свободные грамматики.
Деревья разбора.
Нормальная форма Хомского.
Первая лемма о накачке

Ограничения регулярных грамматик

- (синтаксический моноид) Слова лишь конечно различимы относительно правил переписывания
- (префиксные грамматики) Доступ лишь к началу (концу) слова

Что будет, если снять эти условия?

Структура вывода — дерево, а не последовательность.

Рисунок 23 — К параграфу «Ограничения регулярных грамматик»

Определение 24. *Контекстно-свободная грамматика (CFG) — это грамматика* $\langle \Sigma, N, P, S \rangle$*, где правила переписывания* P *имеют вид* $A \to \alpha$*,* $A \in N$*,* $\alpha \in (\Sigma \cup N)^*$*.*

Контекстно-свободные грамматики

- Нетерминалы переписываются независимо друг от друга (можно понимать их как нульместные функции).
- Вывод в грамматике (разбор слова) не линеен.

S	→	S S		S	→	B	R	→	)
S	→	(S)		B	→	(RB	R	→	(RR
S	→	ε		B	→	ε			

Рисунок 24 — К параграфу «Контекстно-свободные грамматики»

S	→	S S
S	→	(S)
S	→	ε

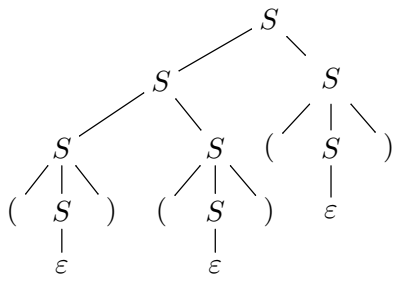


Рисунок 25 — Неоднозначность разбора

Левосторонний разбор

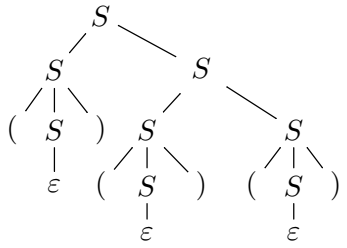
Шаг левостороннего разбора с.ф. $\alpha_1 A \alpha_2$, где $\alpha_1 \in \Sigma^*$, $A \in N$, — замена выделенного вхождения A на правую часть $A \to \beta$. Левосторонний разбор S — разбор, каждый шаг которого левосторонний.

Левосторонний разбор не обязательно единственный, см. ниже.

S → S S S → (S) S → ε

Рисунок 26 — К параграфу «Левосторонний разбор»

Между деревьями разбора слов $w \in L(G)$ и левосторонними разборами w есть взаимно-однозначное соответствие.



S	→	B	R	→	)
B	→	(RB	R	→	(RR
B	→	ε			

Рисунок 27 — (Не)однозначность грамматик

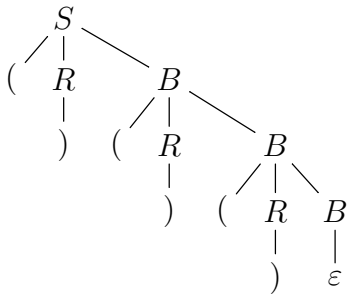


Рисунок 28 — (Не)однозначность грамматик

Грамматика G_2 однозначна — для всех $w \in L(G_2)$ существует единственный левосторонний разбор w. Достаточно заглянуть на 1 символ после разобранной позиции.

Другие проблемы контекстно-свободного разбора слов

- ε-правила (правила вида $A \to \varepsilon$);
- «ε-переходы», или цепные правила (правила вида $A \to B$).

Устранение ε-правил

$A \in N$ коллапсирует, если $A \to \varepsilon \in P$ или $A \to \alpha \in P$ и все элементы α коллапсируют.

- Объявляем `Nullable` $= \emptyset$;

- $\forall A \in N$, если $A \to \varepsilon$, тогда $\texttt{Nullable} = \texttt{Nullable} \cup \{A\}$;
- Пока `Nullable` меняется:
  - для всех $A \in N$, если $A \to B_1 \dots B_n$, $B_i \in \texttt{Nullable} \Rightarrow \texttt{Nullable} = \texttt{Nullable} \cup \{A\}$.
- Итоговое множество `Nullable` — множество всех коллапсирующих нетерминалов.

Предложение 13.
- *Если $\varepsilon \in L(G)$, тогда добавляем новый стартовый символ S_0 и правила $S_0 \to \varepsilon$, $S_0 \to S$.*
- *Стираем все правила $B_i \to \varepsilon$, кроме $S_0 \to \varepsilon$.*
- *Для всех правил $A \to \alpha_1 B_i \alpha_2$, где $B_i \in$ `Nullable`, добавляем правила $A \to \alpha_1\alpha_2$. И получаем новые ε-правила! Порядок преобразований существенен.*

Предложение 14.
- *Если $\varepsilon \in L(G)$, тогда добавляем новый стартовый символ S_0 и правила $S_0 \to \varepsilon$, $S_0 \to S$.*
- *Для всех правил $A \to \alpha_1 B_i \alpha_2$ ($|\alpha_1\alpha_2| \geq 1$), где $B_i \in$ `Nullable`, добавляем правила $A \to \alpha_1\alpha_2$.*
- *Стираем все правила $B_i \to \varepsilon$, кроме $S_0 \to \varepsilon$.*

Построение 12.
- *Строим транзитивное замыкание $A \to_c^* B$ отношения $A \to_c B : A \to B \in P$.*
- *$\forall A, B : A \to_c B$, строим множество правил $A \to \phi_i$, для которых $\exists C, \phi_i (C \to \phi_i \in P \;\&\; (B \to_c^* C \vee C = B) \;\&\; (|\phi_i| > 1 \vee \phi_i = \varepsilon \vee (\phi_i = a \;\&\; a \in \Sigma)))$.*
- *Удаляем все правила $A \to B$.*

Определение 25. *Грамматика G находится в нормальной форме Хомского (CNF) $\Leftrightarrow$ все её правила имеют вид либо $A \to a$, либо $A \to BC$, либо $S \to \varepsilon$, причём S не входит в правую часть никакого правила из G.*

Нормальная форма Хомского

- Устраняем ε-правила.
- Устраняем цепные правила.
- $\forall a \in \Sigma$ таких, что a входит в правую часть правила, отличную от a, заводим нетерминал–охранник G_a, строим правило $G_a \to a$, и во всех правых частях, кроме совпадающих с a, заменяем a на G_a.
- $\forall A \to B_1 \dots B_n$, $n > 2$, вводим новый нетерминал B_{1f} и заменяем $A \to B_1 \dots B_n$ на два правила $A \to B_1 B_{1f}$, $B_{1f} \to B_2 \dots B_n$ (рекурсивно).

Смысл нормальной формы Хомского

1. Неукорачивающие применения правил
2. Нет пустых переходов — правила либо финальные, либо удлиняющие
3. Контролируемый рост длины сентенциальной формы от количества шагов разбора

Перевод грамматики в CNF позволяет легче анализировать свойства её языка и проводить разбор слов.

Недостижимость и зацикливание

- Стартовый нетерминал $S \in N$ достижим.
- Нетерминал $A \in N$ достижим, если существует правило $B \to \alpha$ такое, что $|\alpha|_A \geq 1$ и B достижим.

Предложение 15.
- *Если существует правило $A \to w$, $w \in \Sigma^*$, A порождающий.*
- *Если $A \to \alpha$ и $\forall B_i(|\alpha|_{B_i} \geq 1 \Rightarrow B_i$ порождающий$)$, то A порождающий.*

Недостижимость и зацикливание — продолжение

1. Удаляем из G все правила, в левых или правых частях которых стоят непорождающие нетерминалы.
2. Удаляем из G все правила, в левых или правых частях которых стоят недостижимые нетерминалы.

Проверка корректности рекурсивных алгоритмов

1. Завершаемость — фундированность — искомое множество M нетерминалов не может уменьшаться, и количество нетерминалов грамматики конечно.
2. Корректность — способ доказательства «*minimal bad sequence*» — пусть существуют элементы $k_i \in M$, которые не находятся рекурсивным алгоритмом. Выберем тот из них, до которого минимальный путь из S (варианты — из которого минимальный путь до Σ^*; до ε). Покажем, что есть ещё какой-то с путём вывода ещё короче.

Н.Ф. Хомского и префиксные грамматики

При линеаризации правил, поведение КС-грамматики G в Н.Ф. Хомского в точности описывается алфавитной префиксной грамматикой на нетерминалах.

- Переведём АПГ в регулярную грамматику по методу, описанному в предыдущей лекции.
- У этой грамматики есть длина накачки, т.е. всякое достаточно длинное слово имеет вид $w_1(w_2)^n w_3$, причём $w_1 w_3$ также входит в язык сентенциальных форм G.

Поскольку G — КС-грамматика, то $w_1 \longrightarrow \alpha_1$, $w_3 \longrightarrow \alpha_3$, каждое из $w_2 \longrightarrow \alpha_2$.

Но ещё есть шаг порождения w_2, также выбрасывающий последовательность терминалов.

Лемма 2. *Пусть G — КС-грамматика в форме Хомского. Тогда существует $p \in \mathbb{N}$ такое, что любое слово $w \in L(G)$ длины не меньше p имеет представление вида $x_1 y_1 z\, y_2 x_2$, где $|y_1 y_2| \geq 1$, $|y_1 z\, y_2| \leq p$, и все слова вида $x_1 y_1^k\, z\, y_2^k\, x_2$ также принадлежат $L(G)$.*

Лемма о накачке КС-языков

Пусть в н.ф. Хомского G n нетерминалов. Возьмём $p = 2^n$. Его вывод будет иметь минимум высоту $n + 1 \Rightarrow$ в нём будет существовать путь, содержащий два одинаковых нетерминала A.

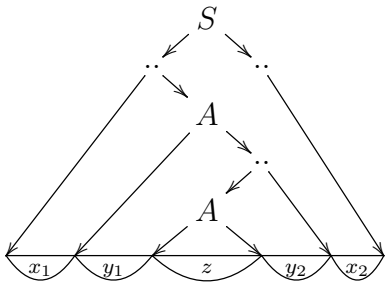


Рисунок 29 — К параграфу «Лемма о накачке КС-языков»

Выберем самые нижние два одинаковых нетерминала $\Rightarrow$ высота поддерева от первого из них не больше $n + 1 \Rightarrow$ длина выводимого слова $y_1 z\, y_2 \leq 2^n$ (т.е. $\leq p$). □

Парсинг в Python

Проанализировать язык $\{a^n z_1 a^n z_2 a^n | n \geq 1, |z_i|_a = 0, |z_i| \geq 1\}$.

Пусть длина накачки есть p. Рассмотрим слово $a^p b a^p b a^p$. Заметим, что если $y_1 z y_2 = a^i b a^j$ (где i и j могут быть равны 0), тогда $|y_1 y_2|_b = 0$. Действительно, иначе нулевая накачка породит слово $a^m b a^p$, которое не принадлежит языку.

Значит, $y_1 = a^j$, $y_2 = a^i$. Однако слова $a^{p+i+j} b a^p b a^p$, $a^p b a^{p+i+j} b a^p$, $a^p b a^p b a^{p+i+j}$, $a^{p+j} b a^{p+i} b a^p$, $a^p b a^{p+j} b a^{p+i}$ ни одно не принадлежат требуемому языку $\Rightarrow$ он не контекстно-свободен.

Теоретико-игровая интерпретация

Достаточное условие непринадлежности языка L к КС по лемме о накачке: $\forall p \exists w \in L(|w| > p \;\&\; \forall x_i, y_i, z(w = x_1 y_1 z y_2 x_2 \;\&\; |y_1 z y_2| < p \Rightarrow \exists i(x_1 y_1^i z y_2^i x_2 \notin L)))$.

В пренексной форме этого условия кванторы образуют последовательность: $\forall\exists\forall\exists$. Эта последовательность задаёт правила игры, где каждый квантор $\exists$ — ход протагониста, квантор $\forall$ — ход антагониста. Ходы антагониста назначают неопределённые параметры. Ходы протагониста дают выбор известной вам структуры, зависящей от ходов антагониста. В случае леммы о накачке это выглядит так.

- Антагонист выбирает длину накачки p.
- Зная p, протагонист выбирает w.
- Антагонист выбирает разбиение w на пять подстрок.
- Возможно, в зависимости от этого разбиения, протагонист предъявляет i, для которого накачка не выполняется.

Иногда такая система анализа свойств, записанных в виде формул с чередующимися кванторами, также называется игрой Элоизы и Абеляра (по буквам, образующим кванторы $\exists$ и $\forall$).

Пока без доказательства: множество КС-языков замкнуто относительно пересечения с регулярными языками.

Предложение 16. *Если в язык L входят подслова произвольной формы из Σ^+, где $|\Sigma| > 1$, тогда, скорее всего, потребуется пересечь L с регулярным языком, чтобы облегчить поиск свидетельства о ненакачиваемости. Пример: язык $\{w_1 w_1 w_2 \,|\, |w_1|_a = |w_2|_a\}$. Пересечение этого языка с $ba^*bba^*bba^*$ гораздо легче поддаётся анализу, поскольку такие слова разбиваются на подходящие w_1 и w_2 однозначно.*

- *Начальная буква b вынуждает w_1 содержать ровно две буквы b. Действительно, если $|w_1|_b = 1$, тогда второе вхождение w_1 должно будет начинаться с b^2, что противоречит выбору w_1.*
- *Последняя буква b навязывает позицию начала w_2.*

Замечание 1. *Если характеристическая функция L содержит предикат отрицания, связывающий две структуры неопределённого размера, в некоторых случаях это приводит к невозможности применения леммы о накачке. В других можно попробовать воспользоваться приёмом «всё включено». Поскольку мы знаем, что длина накачиваемого фрагмента y_1zy_2 меньше p, то выберем w так, чтобы в нём нашлись всевозможные фрагменты такой длины, удовлетворяющие желательному свойству.*

Техника применения

Покажем, что язык $L = \{w \mid w \neq a^{n^2} \ \& \ w \in \{a,b\}^*\}$ не является КС. Для начала заметим, что слова L содержат произвольные подслова в $\{a,b\}^*$, и пересечём L с a^*. Получим $L' = \{a^k \mid k \neq n^2\}$ — если он не КС, то исходный язык также не КС.

- Антагонист выбирает p.
- Наша задача — подобрать такое k, что $\forall p' \exists i, m(p' < p \Rightarrow k + p' * i = m^2)$. То есть включить возможность взятия любого такого p' в наше значение k как конструктивного элемента для построения квадрата числа.
- Возьмём $k = (p!)^2 + 1$. Тогда при любом значении p', меньшем p, можно взять $i = \frac{p!}{p'} * 2$, и получим $k + p' * i = (p! + 1)^2$.

Построение 13. *Покажем, что язык $L = \{ww^R a^n \mid |w|_a = n\}$ не является КС. Опять сначала избавимся от произвольных подслов в L и пересечём его с языком $ba^+b^2a^+ba^+$. Пересечение с таким языком вынуждает w иметь вид ba^ib, а весь язык — вид $L' = \{ba^nbba^nba^n\}$.*

- *Абеляр выбирает p. Элоиза строит слово $ba^pb^2a^pba^p$. Абеляру предоставляется возможность построить его разбиение на $x_1y_1zy_2x_2$.*
- *Если Абеляр выберет $|y_1|_a > 0 \ \& \ |y_1|_b > 0$ (т.е. y_1 содержащим сразу буквы a и b), тогда ненулевая накачка сразу же выведет нас из языка. Аналогично с y_2.*
- *Если Абеляр решит накачивать только b (т.е. выберет y_1 либо y_2 равными b или b^2), тогда любая накачка также будет выводить из языка.*
- *Остаётся только возможность $y_1 = a^i$, $y_2 = a^j$, что позволяет следующие накачки y_1, y_2 на расстоянии не больше p:*
  - $ba^{p+i*k}b^2a^{p+j*k}ba^p$ *— можно сохранить свойство палиндрома, но нельзя сохранить корректный подсчёт букв a, последний индекс не меняется.*
  - $ba^pb^2a^{p+i*k}ba^{p+j*k}$ *— теряется свойство палиндрома.*
  - $ba^{p+(i+j)*k}b^2a^pba^p$ *— теряется свойство палиндрома, при накачке только второго подслова a^p аналогично.*
  - $ba^pb^2a^pba^{p+(i+j)*k}$ *— некорректный подсчёт букв a в w.*

Некоторые не КС-языки тоже накачиваются, например, $\{a^mb^nc^nd^n \mid m > 0\} \cup \{b^ic^jd^k\}$.

Действительно, если слово языка содержит буквы a, тогда мы можем взять $y_1y_2 = a^i$. Иначе накачку можно выбрать произвольно.

То, что этот язык — не КС, можно понять по тому факту, что его пересечение с регулярным языком $ab^*c^*d^*$ не контекстно-свободно.

Пример 8. *Иногда пересечение с регулярным языком делает язык «излишне накачиваемым»: например, пересекая* $L = \{ww^R a^n \mid |w|_a = n\}$ *с* $ba^+b^*a^+ba^+$*, мы даём возможность Абеляру выбрать в качестве* y_1 *пару букв из центрального блока* b^* *(положив* $y_2 = \varepsilon$*). Заметим, что слова без этого блока будут иметь вид* $ba^{2n}ba^n$ *— а такие слова тоже можно накачивать, выбрав* y_1 *из* a^{2n}*,* y_2 *— из* a^n*.*

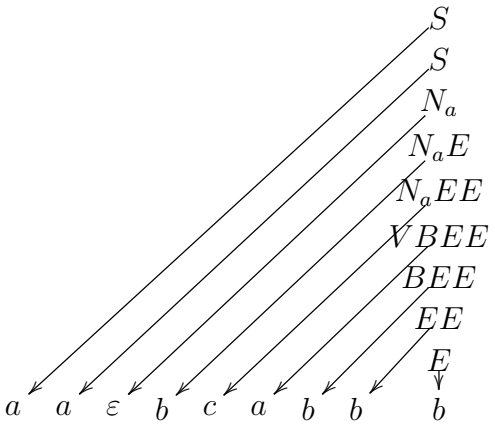
Семейство лемм о накачке. Нормальная форма Грейбах. Теорема Хомского–Шутценберже

Рассмотрим язык сентенциальных форм, но только без учёта терминальных символов. Получим алфавитную префиксную грамматику (АПГ, см. лекцию 4). При этом каждое применение правила переписывания такой грамматики выбрасывает не более чем один терминальный символ слева, а стирающее правило — ровно один.

Связь КС-грамматик и АПГ

Рассмотрим КС-грамматику, соответствующую ей АПГ и путь вывода слова, получаемый с помощью АПГ.

$$\begin{array}{lcl} S & \to & aS \mid bS \mid aN_a \mid bN_b \\ N_a & \to & N_aE \mid bN_aE \mid cVB \\ N_b & \to & aN_bE \mid bN_bE \mid VA \\ V & \to & aV \mid bV \mid a \mid b \\ A \to a & & B \to b \quad E \to a \mid b \end{array} \qquad \begin{array}{lcl} S & \to & S \mid N_a \mid N_b \\ N_a & \to & N_aE \mid VB \\ N_b & \to & N_bE \mid VA \\ V & \to & V \mid \varepsilon \\ A \to \varepsilon & & B \to \varepsilon \quad E \to \varepsilon \end{array}$$



Следствие теоремы Турчина

Пусть G — алфавитная префиксная грамматика, в которой N правил с непустой правой частью, и максимальная длина правой части правила равна M. Тогда любая последовательность порождаемых ею слов

$$\begin{array}{l} a_1 \dots a_n \\ \dots \\ \varepsilon \end{array}$$

длиной не менее $N^M \cdot (n+1)$ содержит пару вида $\tau_1 = \Phi\Theta_0$, $\tau_2 = \Phi\Psi\Theta_0$, такую, что $|\Phi| \le M$ и на отрезке $[\tau_1, \tau_2]$ нет слов длины меньше $|\Theta_0| + 1$.

Классическая лемма о накачке

Пусть G — КС-язык. Тогда существует $p \in \mathbb{N}$ такое, что любое слово $w \in L(G)$ длины не меньше p имеет представление вида $x_1y_1z\,y_2x_2$, где $|y_1y_2| \ge 1$, $|y_1z\,y_2| \le p$, и все слова вида $x_1y_1^k\,z\,y_2^k\,x_2$ также принадлежат $L(G)$.

Доказательство: при левостороннем разборе выбираем самую последнюю пару сентенциальных форм.

Вторая лемма о накачке
Пусть G — КС-язык. Тогда существует $p \in \mathbb{N}$ такое, что любое слово $w \in L(G)$ длины не меньше p имеет представление вида $x_1y_1z\,y_2x_2$, где $|y_2| \geq 1$, $|x_1y_1| \leq p$, $|y_2| \leq p$ либо y_2 накачивается отдельно, $|x_2| \leq p$ либо x_2 накачивается отдельно, и все слова вида $x_1y_1^k\,z\,y_2^k\,x_2$ также принадлежат $L(G)$.

Варианты лемм о накачке

- Хотелось бы сдвигать начало отрезка накачки вперёд на любое константное количество букв, аналогично — конец отрезка накачки (используя свойство реверса).
- Понятие накачки может быть применено рекурсивно к некоторым достаточно длинным подсловам выбранного слова. Т.е. можно выкидывать из слова подслова, накачиваемые отдельно, без риска выйти из языка.

Для первого нужна гарантия того, что если порождается k букв слова, то длина сентенциальных форм увеличивается не более чем в $f(k)$ раз (где f — хотя бы полином).

Бонус: лемма Огдена
Пусть L — КС-язык. Тогда существует такое число n, что в любом слове w, $|w| \geq n$, при отметке n или более букв, w представляется в виде $x_1y_1z\,y_2x_2$, причём либо во всех трех из x_1, y_1, z есть отмеченные буквы, либо они есть во всех трех из z, y_2, x_2, в слове y_1zy_2 отмечено не более n букв, и $\forall k(x_1y_1^k\,z\,y_2^k\,x_2 \in L)$.
Исследуем «плохой» язык $\{a^mb^nc^nd^n \mid m > 0\} \cup \{b^ic^jd^k\}$ с помощью леммы Огдена. Абеляр (антагонист) выбирает n. Элоиза (т.е. мы) строит слово $ab^{2n}c^{2n}d^{2n}$ и отмечает n последних букв d. Абеляр может разбить слово $ab^{2n}c^{2n}d^{2n}$ на $x_1y_1zy_2x_2$ двумя способами:

- отмечены x_1, y_1, z, накачиваться может только d^{2n}.
- отмечены x_2, y_2, z, накачивается либо d^{2n}, либо d^{2n} совместно с c^{2n}, b^{2n} или a.

Оба типа накачки выводят из языка, поскольку при любой положительной накачке число вхождений букв b или c расходится с числом вхождений d в слово.

Н.ф. Хомского и левосторонний вывод

- Могут быть непродуктивные левосторонние цепочки: $A \to AB \to \ldots AB^n \to \ldots$.
- Есть гарантия роста слова при развёртке, но нет определённости, по какому префиксу.

Определение 26. *Грамматика G ($\varepsilon \notin L(G)$) находится в GNF (н.ф. Грейбах) $\Leftrightarrow$ каждое её правило имеет вид $A_i \to a_j\alpha$, где $A_i \in N$, $\alpha \in N^*$, $a_j \in \Sigma$.*

Нормальная форма Грейбах

- Левосторонний разбор по грамматике в GNF на каждом шагу переписывания порождает терминальный символ.
- Для приведения к GNF нужно «вытащить из рекурсии» возможные first-терминалы, порождаемые нетерминалами грамматики.
  - явно найти все завершающиеся цепочки вывода;
  - рассмотреть язык-реверс сентенциальных форм.
- По умолчанию считаем, что к GNF приводится CNF (н.ф. Хомского).

Предложение 17. *1. Нумеруем нетерминалы в правых частях правил в порядке их вхождения;*

2. *(по исчерпанию по i, начиная с $i = 1$) Если имеется правило вида $A_i \to B_j\beta$, где $j < i$, тогда подставляем вместо B_j все правые части α_k правил вида $B_j \to \alpha_k$.*

3. *Если после этого все правила имеют вид либо $A_i \to a\alpha$, $a \in \Sigma$, либо $A_i \to B_j\beta$, причём $i < j$, тогда GNF получается последовательной развёрткой B_j. Существует лексикографический порядок на функциональных символах из N.*

4. *Если есть правила вида $A_i \to A_i\alpha$, тогда устраняем левую рекурсию.*

5. *Увеличиваем i, если ещё остались нетерминалы, не приведённые к ГНФ.*

Устранение левой рекурсии

1. Предположим, для A_i нашлось n леворекурсивных правил и m упорядоченных лексикографически:

$$\begin{array}{ll} A_i \to A_i\alpha_1 & A_i \to \beta_1 \\ \dots & \dots \\ A_i \to A_i\alpha_n & A_i \to \beta_m \end{array}$$

2. Вводим новый нетерминал A'_i такой, что его вес меньше всех прочих, и заменяем правила на:

$$\begin{array}{ll} A'_i \to \alpha_1 A'_i \,|\, \alpha_1 & A_i \to \beta_1 \,|\, \beta_1 A'_i \\ \dots & \dots \\ A'_i \to \alpha_n A'_i \,|\, \alpha_n & A_i \to \beta_m \,|\, \beta_m A'_i \end{array}$$

3. После всех таких замен грамматика лексикографически упорядочена по левому разбору, и GNF получается последовательной левой развёрткой.

Алгоритм Блюма–Коха (1999).

Построение 14. • *Рассмотрим язык сентенциальных форм с переписыванием только по левому разбору. Он регулярен, и в конечное состояние его НКА ведут стрелки, помеченные терминалами.*

• *Для такого языка легко построить инверсный $\Rightarrow$ множество терминалов-префиксов, которые может породить данный нетерминал.*

Предложение 18. *1. По каждому нетерминалу B строим автомат $M_B = \langle N_B \cup \{S_B\}, \Sigma \cup N, B_B, \{S_B\}, \delta\rangle$ (S_B — новое состояние, N_B — множество нетерминалов CFG, индексированное нетерминалом B). Правила перехода δ:*

- $\langle C_B, E, M\rangle \Leftrightarrow M = \{D_B \,|\, C \to DE \in P\}$;
- $\langle C_B, a, \{S_B\}\rangle \Leftrightarrow C \to a \in P$.

2. *Строим реверс к M_B, получаем НКА M_B^R.*

3. *Строим грамматику $G'_B = \langle N_B \cup \{S_B\}, \Sigma \cup N, R'_B, S_B\rangle$ для M_B^R с правилами переписывания:*

- $S_B \to aC_B \Leftrightarrow \langle S_B, a, C_B\rangle \in \delta^R$ *и* $C_B \neq B_B$ *либо из* B_B *есть стрелки в* M_B^R;
- $S_B \to a \Leftrightarrow \langle S_B, a, B_B\rangle \in \delta^R$;
- $D_B \to EC_B \Leftrightarrow \langle D_B, E, C_B\rangle \in \delta^R$ *и* $C_B \neq B_B$ *либо из* B_B *есть стрелки в* M_B^R;
- $C_B \to E \Leftrightarrow \langle C_B, E, B_B\rangle \in \delta^R$.

Определение 27. *Теперь по всем G'_i строим окончательный вариант грамматики $G_B = \langle N_B \cup \{S_B\}, \Sigma, R_B, S_B\rangle$ с правилами:*

- $S_B \to aC_B,\ S_B \to aC_B \in R'_B$;
- $S_B \to a\ S_B \to a \in R'_B$;
- $D_B \to \alpha C_B \Leftrightarrow D_B \to EC_B \in R'_B\ \&\ S_E \to \alpha$ *(по всем таким α и E);*
- $D_B \to \alpha \Leftrightarrow D_B \to E \in R'_B\ \&\ S_E \to \alpha$ *(по всем таким α и E).*

Грамматика $\bigcup\limits_{i\in N} G_i$ со стартовым символом S_S — это искомая GNF для исходной грамматики G.

Последние два правила разворачивают неразмеченные нетерминалы в стартовые правила грамматик их сентенциальных форм, поэтому автоматы M_B имеет смысл строить только для тех нетерминалов, которые встречаются в исходной грамматике не только первыми в правых частях правил.

Пример преобразования грамматики по Блюму–Коху
Привести к GNF грамматику некорректных сумм двоичных чисел (почему некорректных?)

$$S \to S + S \mid D \qquad D \to D0 \mid D1 \mid 1 \mid (S)$$

Сначала избавляемся от цепного правила $S \to D$. Потом строим порождающую структуру A_V сентенциальных форм по левостороннему разбору с финальным состоянием N_V и стартовым V_V. Каждому нетерминалу V соответствует своя структура.

$$\begin{array}{l}\text{Для } A_S: \quad S_S \xrightarrow{+S} S_S \quad S_S \xrightarrow{0} D_S \quad S_S \xrightarrow{1} D_S \quad S_S \xrightarrow{1} N_S \\ S_S \xrightarrow{(S)} N_S \quad D_S \xrightarrow{0} D_S \quad D_S \xrightarrow{1} D_S \quad D_S \xrightarrow{1} N_S \quad D_S \xrightarrow{(S)} N_S\end{array}$$

$$\text{Для } A_D: \quad D_D \xrightarrow{0} D_D \quad D_D \xrightarrow{1} D_D \quad D_D \xrightarrow{1} N_D \quad D_D \xrightarrow{(S)} N_D$$

Превращаем структуры в праволинейные (меняя местами нетерминалы левых и правых частей правил и стартовые состояния с финальными):

$$\begin{array}{l}\text{Для } A_S: \quad S_S \xrightarrow{+S} S_S \quad D_S \xrightarrow{0} S_S \quad D_S \xrightarrow{1} D_S \quad N_S \xrightarrow{1} S_S \\ N_S \xrightarrow{(S)} S_S \quad D_S \xrightarrow{0} D_S \quad D_S \xrightarrow{1} D_S \quad N_S \xrightarrow{1} D_S \quad N_S \xrightarrow{(S)} D_S\end{array}$$

$$\text{Для } A_D: \quad D_D \xrightarrow{0} D_D \quad D_D \xrightarrow{1} D_D \quad N_D \xrightarrow{1} D_D \quad N_D \xrightarrow{(S)} D_D$$

Извлекаем праволинейные (почти) грамматики. В правой части правила может не быть нетерминалов, если там стоял нетерминал V_V.

$$\begin{array}{llllll}\text{q-RLG } G_S: & S_S \to +SS_S & D_S \to 0S_S & D_S \to 1D_S & N_S \to 1S_S \\ & N_S \to (S)S_S & D_S \to 0D_S & D_S \to 1D_S & N_S \to 1D_S & N_S \to (S)D_S \\ & S_S \to +S & D_S \to 0 & D_S \to 1 & N_S \to 1 & N_S \to (S) \\ \text{q-RLG } G_D: & D_D \to 0D_D & D_D \to 1D_D & N_D \to 1D_D & N_D \to (S)D_D \\ & D_D \to 0 & D_D \to 1 & N_D \to 1 & N_D \to (S)\end{array}$$

Заменяем неразмеченные нетерминальные символы V исходной грамматики на N_V. В данном случае нет правил, в которых неразмеченные нетерминалы стояли бы первыми в правых частях, поэтому достаточно просто заменить их на N_V. Иначе пришлось бы заменять их на все возможные правые части α правил вида $N_V \to \alpha$. Стартовый символ — N_S. GNF почти построена!

$$\begin{array}{llllll}\text{q-GNF для } G: & S_S \to +N_SS_S & D_S \to 0S_S & D_S \to 1D_S & N_S \to 1S_S \\ & N_S \to (N_S)S_S & D_S \to 0D_S & D_S \to 1D_S & N_S \to 1D_S & N_S \to (N_S)D_S \\ & S_S \to +N_S & D_S \to 0 & D_S \to 1 & N_S \to 1 & N_S \to (N_S)\end{array}$$

Осталось обернуть в delay-нетерминалы терминальные символы правых частей правил, кроме первого. Здесь это символ).

Пусть $PAREN_n$ — язык из $4 * n$ элементов $\{[_1,]_1, \ldots, [_n,]_n, (_1,)_1, \ldots, (_n,)_n\}$.

Теорема 5. *Любой CF-язык получается гомоморфизмом из языка $L' = PAREN_n \cap R$, где R — регулярный.*

Теорема Хомского–Шутценберже

Пусть G — грамматика L в нормальной форме Хомского. Пронумеруем правила G и поставим им в соответствие следующие.

1. Если правило n имеет вид $A \to BC$, тогда порождаем правило $A \to [_n B]_n (_n C)_n$.
2. Если правило n имеет вид $A \to a$, тогда порождаем правило $A \to [_n]_n (_n)_n$.

Свойства языка $L(G')$

- Все $]_n$ строго предшествуют $(_n$.
- Ни одна $)_n$ не предшествует непосредственно левой скобке.
- Если правило n — это $A \to BC$, тогда $[_n$ непосредственно предшествует некоторой $[_p$, так же как и $(_n$.

Язык R

$R = \{x \in \{[_j,]_j, (_j,)_j\}^* \mid x$значинается с $[_n$для некоторого правила $n : A \to \cdots$ & все $]_n$предшествуют $(_n\}$. Можно убедиться, что $L(G') = R \cap PAREN_n$.

Осталось определить h. Если n — нефинальное правило, то $h([_n) = h(]_n) = h((_n) = h()_n) = \varepsilon$. Иначе $h([_n) = a$, для остальных скобок так же.

Значение теоремы Х.-Ш.

Возможно разделить парсинг любого КС-языка на две стадии: лексический анализ (проверка условия R) и разбор правильных скобочных структур.

Замечание: поскольку гомоморфизм h не обязан быть инъективным, разбор ПСП не всегда можно определить однозначно. Пример: $\{a^n b^n\} \cup \{a^n b^{2n}\}$ (полностью неоднозначность устранить нельзя, т.к. этот язык не является детерминированным). Однако Т.Х.Ш. даёт подсказку, как строить КС-грамматики: надо найти в языке все скрытые «скобочные структуры».

Построение 15. *Построить КС-грамматику для языка $\{a^n b^m c^k \mid n = 2 * m - k\}$.*

Ищем возможную скобочную структуру. Для этого сначала избавимся от вычитания: $n + k = 2 * m$. Значит, буквы a должны балансироваться буквами b справа (т.е. буквы b являются «закрывающими скобками» для a), а буквы c — буквами b слева (т.е. буквы b являются «открывающими» для c). Возможны два случая: n и k оба чётны либо оба нечётны. Построим соответствующие им разбиения: $\{a^{2*n'} b^{n'} b^{k'} c^{2*k'}\}$ и $\{a a^{2*n'} b^{n'} b b^{k'} c^{2*k'} c\}$. Дальнейшее построение грамматики уже очевидно. Заметим, что гомоморфизм подразумевает минимум четыре вида скобок: пара $(_{2a},)_b$, пара $(_b,)_{2c}$, внешняя пара $[_a,]_c$ (для нечётного варианта) и $[_b,]_\varepsilon$ для него же, чтобы породить внутреннюю букву b. Как итог, получаем язык, гомоморфно порождаемый языком Дика над $\{(_{2a},)_b, (_b,)_{2c}, [_b,]_\varepsilon, [_a,]_c\}$ со следующим лексером:

1. До $(_{2a}$ может идти лишь единственная $[_a$.
2. После $)_b$ распознаётся одна $[_b$, если распозналась $[_a$.
3. После $)_b$ или $]_\varepsilon$ не может идти ничего другого, кроме $(_b$ или $]_c$ (последняя — только после $]_\varepsilon$).

4. После $)_{2c}$ не может быть ничего, кроме $)_{2c}$ или $]_c$. Дополнительное условие на существование $[_a$ уже не требуется — оно следует из сбалансированности ПСП.

Конструкция выше отличается от используемой в доказательстве теоремы — в целях экономии, в ней почти нет скобок, гомоморфно отображаемых в пустое слово.

Построение 16. *Построить КС-грамматику для* $L_{\neq} = \{w_1 c w_2 \mid w_i \in \{a,b\}^+ \;\&\; w_1 \neq w_2\}$.

Классический пример грамматики с не-КС дополнением. Чтобы расшифровать неравенство, раскроем его в дизъюнкцию: «слово w_1 короче, чем w_2; либо w_2 короче, чем w_1; либо существует такое i, что w_1 и w_2 различаются в i-й позиции». Здесь условия не взаимоисключающие: достаточно одного из них, чтобы слово принадлежало $L_{\neq}$, но могут выполняться и два сразу.

Перепишем первое условие: $w_1 c w_1' w_2$, где $|w_2| > 0$ и $|w_1| = |w_1'|$. Очевидно, что «открывающими скобками» будут буквы из w_1, «закрывающими» — из w_1', а «скобки» для w_2 замкнуты на самом w_2. Чтобы обеспечить четыре вида соответствий букв по счёту, придётся ввести четыре пары скобок для w_1 и w_1': $\{(_a,)_a, (_b,)_b, [_a,]_b, [_b,]_a\}$. И две пары скобок для w_2: $\{\{_a, \}_\varepsilon, \{_b', \}_\varepsilon'\}$ и пара скобок для порождения c: $(_c$ и $)_\varepsilon$(в нижних индексах — гомоморфные образы скобок).

Осталось построить регулярные условия. Для языка $w_1 c w_1' w_2$, где $|w_2| > 0$ и $|w_1| = |w_1'|$, их можно описать следующим образом: **Дополнительный пример**

1. После $(_a$, $(_b$, $[_a$, $[_b$ всегда идёт либо опять одна из таких скобок, либо $(_c$. Скобка $(_c$ единственна.

2. После $)_\varepsilon$, а также скобок $)_a$, $)_b$, $]_b$, $]_a$, могут идти либо $)_a$, $)_b$, $]_b$, $]_a$, либо фигурные скобки.

3. В каждом слове есть хотя бы одна фигурная скобка. После открывающей фигурной скобки обязательно сразу идёт закрывающая, и после первой встреченной фигурной скобки все остальные скобки — тоже фигурные.

Представление Хомского–Шутценбергера для языка $w_1 w_2 c w_1'$, где $|w_2| > 0$ и $|w_1| = |w_1'|$, строится симметрично.

Осталось разобрать $\{w_1 t_1 w_2 c w_3 t_2 w_4 \mid |w_1| = |w_3| \;\&\; t_1 \neq t_2\}$. Очевидно, что в нём «открывающими» будут элементы w_1, закрывающими — элементы w_3, t_1 и t_2 — уникальные скобки, отображающиеся в разные элементы алфавита, а w_2 и w_4 закрываются сами собой (как w_2 в предыдущем языке). Для $t_1 + t_2$-скобок назначим пары $[_a^t$, $]_b^t$ и $[_b^t$, $]_a^t$, остальные обозначения сохраним те же. Лексер языка:

1. После $(_a$, $(_b$, $[_a$, $[_b$ идёт либо опять одна из таких скобок, либо $[_\_^t$-скобка. $[_\_^t$ единственна, за ней следует либо $\{_a$, либо $\{_b'$, либо $(_c$.

2. За открывающей фигурной скобкой следует закрывающая, и после первой встреченной фигурной скобки все остальные скобки — тоже фигурные, до конца строки либо до чтения единственной $(_c$.

3. После $)_\varepsilon$, а также скобок $)_a$, $)_b$, $]_b$, $]_a$, могут идти либо $)_a$, $)_b$, $]_b$, $]_a$, либо скобка $]_\_^t$. За $]_\_^t$ следует EOL или $\{_a$ или $\{_b'$.

Чтобы получить прообраз языка $L_{\neq}$, объединим все три лексера.

Теорема Париха. Стековые автоматы

Теорема Париха
Скажем, что множество векторов полулинейно, если оно является конечным объединением множеств векторов вида $(i_1 + \sum p_{t,1} \cdot j_{t,1}, \ldots, i_k + \sum p_{t,k} \cdot j_{p,k})$.

КС-язык $\mathscr{L}$ в алфавите Σ можно описать количественно: как множество $V_{\mathscr{L}}$ векторов $\{(k_{j,1}, \ldots, k_{j,n}) \mid k_{j,i} = |w_j|_{a_i} \ \& \ w_j \in \mathscr{L}\}$.

Предложение 19. *Пусть $V_{\mathscr{L}} = \{(k_{j,1}, \ldots, k_{j,n}) \mid k_{j,i} = |w_j|_{a_i} \ \& \ w_j \in \mathscr{L}\}$. Если $\mathscr{L}$ — КС-язык, тогда $V_{\mathscr{L}}$ полулинейно.*

Теорема Париха — продолжение

- Если $V_{\mathscr{L}}$ полулинеен, то $V_{h(\mathscr{L})}$ полулинеен (h — гомоморфизм).
- Если w принадлежит языку Шютценберже, то $\forall n(|w|_{(_n} = |w|_{)_n} = |w|_{[_n} = |w|_{]_n})$.
- Рассматриваем префиксные трассы вывода над ГНФ языка Шютценберже и выкидываем из них наиболее короткие отрезки накачки. Их длина ограничена $\Rightarrow$ ограничено их множество.

Следствия теоремы Париха
Множества регулярных и КС-языков над однобуквенным алфавитом совпадают.

Коммутативным образом всякого КС-языка является регулярный язык.

Пусть стартовый нетерминал грамматики G_i — это S_i.

Свойства замкнутости КС-языков

- КС-языки тривиально замкнуты относительно объединения и конкатенации. Объединение: добавим правило $S' \to S_1 \mid S_2$, конкатенация: добавим правило $S' \to S_1 S_2$.
- КС-языки замкнуты относительно реверсирования (достаточно реверсировать левые части правил), а также префиксных и суффиксных замыканий (упражнение после освоения материала по PDA).
- КС-языки не замкнуты относительно пересечения. Универсальный контрпример: пусть \$ – символ-разделитель (отсутствует в словаре). Рассмотрим язык $\{w_1\$(.)^*\$w_3\}$, где слова w_1 и w_2 связаны порождающими правилами (т.е. структура $w_1\$w_3$ не регулярна), и язык $\{w_1\$w_2\$(.)^*\}$, где такими правилами связаны w_1 и w_2. Их пересечение вынудит существование нерегулярной зависимости между w_1 и одновременно w_2 и w_3, что порождает не две, как в КС-языках, а минимум три области накачки.

  Конкретные примеры: пара $L_1 = \{a^n\$a^n\$a^*\}$, $L_2 = \{a^n\$a^*\$a^n\}$; или пара $L_1' = \{w\$w^R\$a^*\}$, $L_2' = \{w\$(a|b)^*\$a^n \mid |w|_a = n\}$.
- КС-языки не замкнуты относительно дополнения. В противном случае пересечение также не нарушало бы свойство контекстной свободы. Контпримеры строятся на основе этой же идеи: берём язык-пересечение КС-языков L_1 и L_2, не являющийся КС. Чаще всего его дополнение — КС-язык.

  Конкретные примеры: КС-язык $L' = \overline{L_1 \cap L_2} = \{a^m\$a^n\$a^k \mid m \neq n \vee m \neq k \vee n \neq k\}$ с не КС-дополнением; КС-язык $L'' = \overline{L_1' \cap L_2'} = \{w\$v\$a^n \mid v \neq w^R \vee n \neq |w|_a\}$ с не КС-дополнением.
- КС-языки замкнуты относительно пересечения с регулярным языком (см. ниже).

Построение 17. *Даны КС-грамматика G и конечный автомат $\mathscr{A}$. Можно построить КС-грамматику G' такую, что $L(G') = L(G) \cap L(\mathscr{A})$.*

Предположим, что G — в k-нормальной форме Хомского (т.е. с максимум k нетерминалами в правых частях), q — множество состояний автомата $\mathscr{A}$, q_f — единственное финальное состояние, N — множество нетерминалов грамматики G. Множество нетерминалов G' — множество $\langle q_i, A, q_j \rangle$, $q_i, q_j \in q$, $A \in N$.

Пересечение КС-грамматики и рег. языка

- По каждому правилу $A \to A_1 \dots A_n$ из G строим правила $\langle p, A, q\rangle \to \langle p, A_1, q_1\rangle\langle q_{n-1}, A_n, q\rangle$ для всех возможных p, q, q_i.
- По правилу вида $A \to t$ из G и переходу $p \to^t q$ строим правило $\langle p, A, q\rangle \to t$.
- Нетерминал $\langle q_0, S, q_f\rangle$ объявляем стартовым.

Построение 18. *Построим пересечение языков CFG* $S \to G_A T \mid SS$, $T \to b \mid SG_B$, $G_A \to a$, $G_B \to b$, *и следующего автомата:*

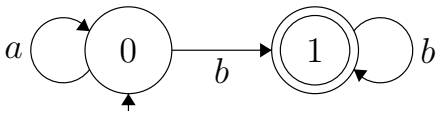


Рисунок 30 — Пример

Сначала разберёмся с правилами вида $X \to t$. Если $t = a$, тогда подходящий нетерминал — только G_A, состояния — только 0+0. Если $t = b$, получается четыре комбинации состояний и нетерминалов.

$\langle 0, G_B, 1\rangle \to b$ $\langle 1, G_B, 1\rangle \to b$
$\langle 0, T, 1\rangle \to b$ $\langle 1, T, 1\rangle \to b$ $\langle 0, G_A, 0\rangle \to a$

$\langle 0, G_A, 0\rangle \to a$ $\quad$ $\langle 1, G_B, 1\rangle \to b$
$\langle 0, T, 1\rangle \to b$
$\langle 0, S, 1\rangle \to \langle 0, G_A, 0\rangle\langle 0, T, 1\rangle$ $\quad$ $\langle 0, T, 1\rangle \to \langle 0, S, 1\rangle\langle 1, G_B, 1\rangle$

$\langle 0, S, 0\rangle \to \langle 0, G_A, 0\rangle\langle 0, T, 0\rangle$ $\quad$ $\langle 0, S, 1\rangle \to \langle 0, G_A, 0\rangle\langle 0, T, 1\rangle$
$\langle 0, T, 1\rangle \to \langle 0, S, 0\rangle\langle 0, G_B, 1\rangle$ $\quad$ $\langle 0, T, 1\rangle \to \langle 0, S, 1\rangle\langle 1, G_B, 1\rangle$
$\langle 1, T, 1\rangle \to \langle 1, S, 0\rangle\langle 0, G_B, 1\rangle$ $\quad$ $\langle 1, T, 1\rangle \to \langle 1, S, 1\rangle\langle 1, G_B, 1\rangle$
$\quad$ $\langle 0, S, 1\rangle \to \langle 0, G_A, 0\rangle\langle 0, T, 1\rangle$
$\langle 0, T, 1\rangle \to \langle 0, S, 0\rangle\langle 0, G_B, 1\rangle$ $\quad$ $\langle 0, T, 1\rangle \to \langle 0, S, 1\rangle\langle 1, G_B, 1\rangle$
$\langle 1, T, 1\rangle \to \langle 1, S, 0\rangle\langle 0, G_B, 1\rangle$ $\quad$ $\langle 1, T, 1\rangle \to \langle 1, S, 1\rangle\langle 1, G_B, 1\rangle$
$\quad$ $\langle 0, S, 1\rangle \to \langle 0, G_A, 0\rangle\langle 0, T, 1\rangle$
$\langle 0, T, 1\rangle \to \langle 0, S, 0\rangle\langle 0, G_B, 1\rangle$ $\quad$ $\langle 0, T, 1\rangle \to \langle 0, S, 1\rangle\langle 1, G_B, 1\rangle$
$\quad$ $\langle 1, T, 1\rangle \to \langle 1, S, 1\rangle\langle 1, G_B, 1\rangle$
$\quad$ $\langle 0, S, 1\rangle \to \langle 0, G_A, 0\rangle\langle 0, T, 1\rangle$
$\langle 0, T, 1\rangle \to \langle 0, S, 0\rangle\langle 0, G_B, 1\rangle$ $\quad$ $\langle 0, T, 1\rangle \to \langle 0, S, 1\rangle\langle 1, G_B, 1\rangle$

$\quad$ $\langle 0, S, 1\rangle \to \langle 0, G_A, 0\rangle\langle 0, T, 1\rangle$
$\quad$ $\langle 0, T, 1\rangle \to \langle 0, S, 1\rangle\langle 1, G_B, 1\rangle$

Рассмотрим возможные подстановки состояний в правила, порождаемые $S \to G_A T$, $T \to SG_B$. Соответствующие уравнения:

$\langle X1, S, X2\rangle \to \langle X1, G_A, X3\rangle\langle X3, T, X2\rangle$
$\langle Y1, T, Y2\rangle \to \langle Y1, S, Y3\rangle\langle Y3, G_B, Y2\rangle$

Чтобы правила были порождающими, необходимо положить $X1 = X3 = 0$, $Y2 = 1$. Выпишем все такие правила. Заметим, что получившийся в одном из них нетерминал $\langle 0, T, 0\rangle$ — непорождающий, и удалим это правило.

Осталось разобраться с правилами, порождёнными $S \to SS$. Выпишем их общий вид: $\langle X1, S, X2\rangle \to \langle X1, S, X3\rangle\langle X3, S, X2\rangle$.

Если положить $X1 = 1$, $X2 = 0$, получим саморекурсивное правило $\langle 1, S, 0\rangle \to \alpha_1\langle 1, S, 0\rangle\alpha_2$. Но в построенной части грамматики нет правил вида $\langle 1, S, \dots\rangle \to \beta$. Поэтому нетерминал $\langle 1, S, 0\rangle$ — непорождающий. Удалим правила с его вхождением. Теперь если $X1 = X2 = 1$, то единственный вариант

развёртки $S \to SS$ без участия нетерминала $\langle 1, S, 0\rangle$ будет иметь вид $\langle 1, S, 1\rangle \to \langle 1, S, 1\rangle\langle 1, S, 1\rangle$, так что нетерминал $\langle 1, S, 1\rangle$ тоже непорождающий. Аналогичным образом устанавливаем бесполезность нетерминала $\langle 0, S, 0\rangle$, который обязан ссылаться либо дважды на себя, либо на непорождающий $\langle 1, S, 0\rangle$. Теперь получается, что все варианты раскрытия нетерминала $\langle 0, S, 1\rangle$ по правилу $S \to SS$ включают непорождающие нетерминалы, поэтому никаких других правил в грамматику добавлять не надо. Осталось только удалить правила с недостижимыми нетерминалами $\langle 0, G_B, 1\rangle$, $\langle 1, T, 1\rangle$. Грамматика пересечения языков построена.

Алгоритм Кока–Янгера–Касами (CYK)

Дано слово $w_1 \dots w_n \in \Sigma^+$ и грамматика G в CNF. Проверить, выполнено ли $w \in L(G)$.

Идея алгоритма: переход к более простым задачам порождения подстрок w.

Определим функцию $f(A, i, j)$ (где $i \le j$), возвращающую ответ, можно ли вывести слово $w_i \dots w_j$ из $A \in N$.

- Если $i = j$, тогда $f(A, i, j) = \mathrm{T} \Leftrightarrow A \to w_i \in P$, и $f(A, i, j) = \mathrm{F}$ иначе.
- Если $i < j$, тогда

$$f(A, i, j) = \bigvee_{(A \to BC \in P)} \bigvee_{k=i+1}^{j} (f(B, i, k-1) \,\&\, f(C, k, j)).$$

Пусть G — CFG. Неформально представим, что G — это стековый автомат, где состояния стека — нетерминальные сент. формы, порождаемые G. Скажем, что G распознаёт только слова, соответствующие пустому стеку.

Грамматика и её стек

$S \to aSB \,|\, SS \,|\, \varepsilon \qquad B \to b$

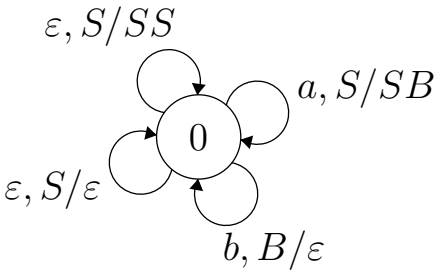


Рисунок 31 — К параграфу «Грамматика и её стек»

А если в такие автоматы добавить ещё состояния?

Определение 28. *Стековый автомат* $\mathscr{A}$ *— кортеж* $\langle \Pi, \Sigma, Q, \delta, q_0, Z_0\rangle$*, где:*

- Π *— алфавит стека;*
- Σ *— алфавит языка;*
- Q *— множество состояний;*
- δ *— правила перехода вида* $\langle q_i, t, P_i\rangle \to \langle q_j, \alpha\rangle$*, где* $t \in \Sigma \cup \{\varepsilon\}$*,* $\alpha \in \Pi^*$*;*
- q_0 *— стартовое состояние,* Z_0 *– дно стека.*

Два варианта допуска слова:

- если слово полностью прочитано, и стек пуст;
- если слово полностью прочитано, и состояние финальное.

Виды допуска

PDA с допуском по конечному состоянию распознают те же языки, что и PDA с допуском по пустому стеку.

- Пусть PDA допускает пустой стек. Добавим новый символ дна Z_1 и добавим по нему ε-переходы из всех состояний в новое финальное состояние.
- Пусть PDA допускает финальные состояния. Добавим из них ε-переходы в состояние, опустошающее стек, а также новый символ стека Z_1 и новое стартовое состояние q_0' с переходом $\langle q_0', \varepsilon, Z_0\rangle \to \langle q_0, Z_0 Z_1\rangle$.

Обычно PDA изображается в виде автомата, в котором стрелки помечены сигнатурой $\alpha,\ T/\Phi$, где α — это символ терминального алфавита (или пустое слово), T — символ на вершине стека, Φ — последовательность стековых символов, помещаемая на вершину стека вместо T.

Пример оформления PDA

Следующий PDA распознаёт правильные скобочные последовательности (включая пустое слово).

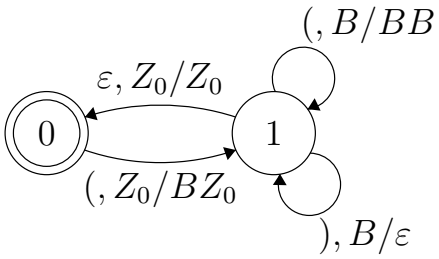


Рисунок 32 — К параграфу «Пример оформления PDA»

Заметим, что перехода из состояния 0 по символу) нет. Так же как и в случае конечных автоматов, можно добавить для такого перехода состояние-ловушку, потому что он порождает слово, в префиксе которого количество закрывающих скобок превышает количество открывающих, а такие слова не являются ПСП.

Построение 19. *По всякой CFG G можно построить PDA $\mathscr{A}$ такой, что $L(G) = L(\mathscr{A})$.*

Построение 20. *Переведём G в GNF и построим по ней PDA с единственным состоянием 0 и допуском по пустому стеку, такой что $Z_0 = S$, правилу $A \to a$ соответствует переход $(0, a, A) \to (0, \varepsilon)$; правилу $A \to aB_1 \dots B_n$ — переход $(0, a, A) \to (0, B_1 \dots B_n)$.*

Построение 21. *По всякому PDA $\mathscr{A}$ можно построить CFG G такую, что $L(G) = L(\mathscr{A})$.*

Построение 22. *Пусть $\mathscr{A}$ допускает слова по пустому стеку.*

- *Построим по стеку $\mathscr{A}$ вспомогательную G':*
  - *введём новые стековые символы и заменим правила $(q_i, t, A) \to (q_j, A_1 \dots A_n)$ $(n \geq 1)$ на пары $(q_i, \varepsilon, A) \to (q_i, A_0 \dots A_n)$, $(q_i, t, A_0) \to (q_j, \varepsilon)$.*
  - *переход $(q_i, \varepsilon, A) \to (q_j, A_0 A_1 \dots A_n)$ поставим в соответствие правилу $A \to A_0 A_1 \dots A_n$; переход $(q_i, t, A) \to (q_j, \varepsilon)$ поставим в соответствие правилу $A \to t_{i,j}$. Z_0 объявим стартовым символом. Пустой символ введём явно и так же пометим.*
- *Построим $\mathscr{A}'$ — FA с правилами вида $(q_i, t_{i,j}) \to q_j$, если для каких-нибудь A, α $(q_i, t, A) \to (q_j, \alpha)$ — переход $\mathscr{A}$. Все состояния объявим финальными.*
- *Теперь построим CFG — пересечение G' и $\mathscr{A}'$ и сотрем все $\varepsilon_{i,j}$ и разметку терминалов. Грамматика G готова!*

PDA в CFG формально

- Нетерминалы — тройки $[p, A, q]$, где $p, q \in Q$, $A \in \Pi$.
- По каждому переходу вида $(q, t, A) \to (p, A_1 \dots A_n)$ добавим правила для всех возможных q_i вида $[q, A, q_n] \to t[p, A_1, q_1] \dots [q_{n-1}, A_n, q_n]$.
- По каждому переходу вида $(q, t, A) \to (p, \varepsilon)$ добавим правило $[q, A, p] \to t$.
- Разрешим стартовому состоянию переписываться в любое из $[q_0, Z_0, q]$.

Определение 29. *PDA $\mathscr{A}$ детерминированный, если:*

- *если есть переход $\langle q, \varepsilon, Z\rangle \to \dots$, то больше никаких переходов по Z из состояния q нет;*
- *каждой тройке $\langle q, a, Z\rangle$, $a \in \Sigma$, соответствует не больше одной правой части.*

Пример 9. *DPDA слабее, чем NPDA. Например, язык $\{a^n b^m \mid n = m \vee m = 2 * n\}$ не распознается DPDA. DPDA с допуском по пустому стеку ещё слабее — язык $\{a^n\}$ не может быть распознан DPDA с таким допуском.*

Предположим, что существует DPDA, распознающий язык $\{a^n b^m \mid n = m \vee m = 2 * n\}$. Тогда после чтения префикса $a^n b^n$ слова $a^n b^{2n}$ он должен находиться в финальном состоянии. Далее он должен распознать ровно n букв b. Заменим часть автомата, распознающую этот фрагмент слова, на изоморфную ей, но читающую только буквы c. Получим PDA, распознающий язык $\{a^n b^n\} \cup \{a^n b^n c^n\}$, не являющийся КС.

Предположим, что существует DPDA с допуском по пустому стеку, распознающий язык $\{a^n\}$. Тогда на слове a стек этого автомата должен быть уже точно пуст $\Rightarrow$ в этом состоянии вообще невозможно сделать дальнейшие переходы.

Пример DPDA

PDA для ПСП, приведённый выше, является DPDA, в чём нетрудно убедиться, проверив, что ε-переход совершается лишь в том случае, когда никакие другие совершить невозможно. Добавим в него состояние-ловушку. Чтобы не описывать многочисленные переходы из состояния-ловушки в себя по всем парам «символ ленты — символ стека», мы воспользовались сокращённым обозначением $\forall$, $\forall/\forall$, подразумевая следующее: «по любой паре $\langle$ терминал, символ стека $\rangle$ в состоянии 2 переходим в себя, сохраняя символ стека на вершине». Также избавимся от ε-перехода, введя символ стека F, т.е. «самая первая скобка». Поскольку автомат $\mathscr{A}$ — детерминированный, в нём существуют переходы по всем комбинациям $\langle$ терминал, символ стека $\rangle$, и нет ε-переходов, связывающих нефинальное и финальное состояния, то автомат, в котором все конечные состояния $\mathscr{A}$ заменены на нефинальные и наоборот, распознаёт дополнение языка, распознаваемого PDA $\mathscr{A}$. Значит, мы показали, что дополнение языка ПСП контекстно-свободно, и предъявили PDA, который распознаёт его.

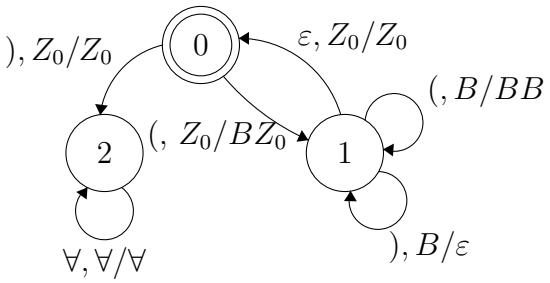


Рисунок 33 — К параграфу «Пример DPDA»

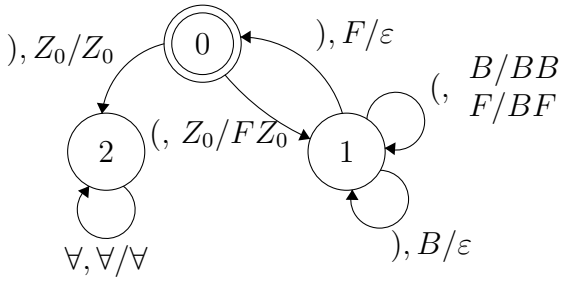


Рисунок 34 — К параграфу «Пример DPDA»

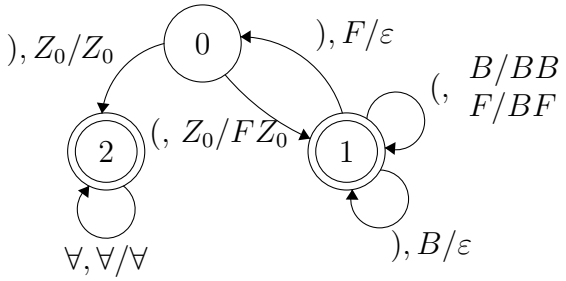


Рисунок 35 — К параграфу «Пример DPDA»

Двухсторонние PDA

Двухсторонние PDA распознают больше языков, чем односторонние.

Доказательство: язык $\{a^n b^n c^n\}$ распознаваем двухсторонним PDA.

Кодирующие КС-языки
Нисходящий разбор

Рассмотрим путь вывода произвольного слова $a_1 \dots a_n$ в праволинейной грамматике. Он имеет вид $S \to a_1 A_1; A_1 \to a_2 A_2; \dots A_n \to a_n$. Применим к нему обратный гомоморфизм $h(A_i; A_i \to) = \varepsilon$ и сотрём префикс $S \to$, получим искомое слово.

Алфавит: $\Sigma \cup N \cup \{;, \to\}$. Описание языка: $\{S \to a_i(A_i; A_i \to a_j)^*\}$.

Описание языка привязано к множеству нетерминалов в рассматриваемой грамматике, но левые и правые вхождения нетерминалов можно было бы закодировать скобками, по количеству считающими номер нетерминала.

Аналогично — с кодировкой путей вывода в КС-грамматике. Но здесь есть проблема с "перепутанными скобками": в пути $A_1 \to a_1 A_2 A_3; A_2 \to \dots; A_3 \to \dots$ имена A_2 и A_3 состоят в зависимости с соответствующими левыми частями, но перемежаются.

Решение

Поменять местами кодировки для A_2 и A_3 в правых частях.

Здесь ε-free вариант. D — язык сбалансированных скобочных структур над $\{(,), [,]\}$.

Язык Грейбах

$L_0 = \{x_1 c y_1 c z_1 d \dots d x_n c y_n c z_n d \,|\, y_1 \dots y_n \in eD \;\&\; z_i, x_i$не содержат $e \;\&\; y_1 \in e\{(,), [,]\}^* \;\&\; y_{i+1} \in \{(,), [,]\}^*\}$

Предложение 20. *Если L — КС-язык, тогда существует $h \in Hom$ такой, что $h^{-1}(L_0) = L$.*

Построение 23. *Пусть G — грамматика для L в форме Грейбах (Шейлы!). Пронумеруем нетерминалы G так, чтобы стартовый был первым. Построим вспомогательную функцию ξ:*

- *для правил $A_i \to a$ положим $\xi(i) =)]^i)$*
- *для правил $A_i \to aA_{j1} \dots A_{jn}$ положим $\xi(i) =)]^i)([^{jm}(\dots([^{j1}($*
- *если $i = 1$, тогда дополнительно припишем префикс $e([(.$*

Пусть терминалом a начинаются левые части правил $k_1, \dots, k_m$. Тогда $h(a) = c\xi(k_1)c \dots c\xi(k_m)d$.

Лемма 3. *Пусть правила грамматики имеют следующий вид:*

$$T \to W_1 T^{i_1} T \mid \dots \mid W_{k-1} T^{i_{k-1}} T \mid W_k$$

где W_i не содержит T, и степени i_j различны. При этом W_i могут содержать любые регулярные операции над константами и нетерминалами, отличными от T. Коммутативный образ правил для T есть $T = (W_1(W_k)^{i_1} + \dots + W_{k-1}(W_k)^{i_{k-1}})^* W_k$.

Ограничение: меняет местами подвыводы, соединяя те, которые накачиваются отдельно.

Определим оператор минимальной неподвижной точки: скажем, что $L(\mu X.r(X))$ — это объединение языков $r(X)$, $r(r(X))$, …, $r^n(X)$.

Что будет, если добавить в регулярные выражения μ-оператор?

μ-регулярные выражения и РБНФ

μ-оператор по переменным + регулярные операции определяют множество всех КС-языков.

Пример: $\mu y.a(\mu x.axb + y + a)$ определяет грамматику $Y \to a\,X$, $X \to a\,X\,b \mid Y \mid a$.

Неудобно, если язык некоторого нетерминала используется в нескольких правилах (например, $S \to A\,B \mid B\,A$ заставит дважды выписывать μ-выражения для A и B).

В действительности аналогом μ-регулярных выражений выступает РБНФ: в правых частях правил разрешены любые регулярные операции.

Проблемы PDA ~~и сила Рефала~~

Не всегда по стеку и входным символам можно понять, что делать со стеком. Пример: грамматика палиндромов.

Неформально: заглядывание вперёд на произвольное, но заранее ограниченное число символов не даёт никакой информации о том, что делать со стеком.

- Возьмём слова $a^{2p}\,b\,a^{2p}$ и a^{2p}.
- После чтения p символов в первом случае нужно продолжать накапливать стек, а во втором — начинать вынимать из него.
- Но впереди одни и те же a^p букв...

Предложение 21. *Пусть PDA-анализатор хранит пару $\langle \alpha, a_i \dots a_{i+k} \rangle$, где α — сент. форма, $a_i \dots a_{i+k}$ — k следующих символов в строке.*

- *Если $\alpha = A\alpha'$, тогда по правилу $A \to \beta$ стековый анализатор переходит в $\langle \beta\alpha', a_i \dots a_{i+k} \rangle$, либо сообщает об ошибке.*
- *Если $\alpha = a_i\alpha'$, тогда a_i одновременно снимается со стека и читается во входной строке.*

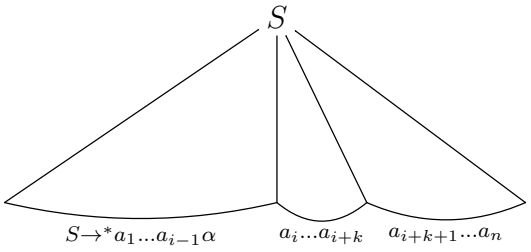


Рисунок 36 — Нисходящий разбор

Определение 30. *Грамматика G называется LL(k) (left-to-right, leftmost derivation) $\Leftrightarrow$ в ситуации, когда существуют выводы $S \to^* w_1 A\alpha \to w_1\xi\alpha \to^* w_1cw_2$, $S \to^* w_1 A\alpha \to w_1\eta\alpha \to^* w_1cw_3$, причём $c \in \Sigma^k$, $w_1, w_2, w_3 \in \Sigma^*$, или $c \in \Sigma^{<k}$, $w_2 = w_3 = \varepsilon$, всегда $\eta = \xi$.*

Неформально: неоднозначность в выборе правила грамматики при разборе сверху вниз устраняется заглядыванием вперёд на k букв.

Предложение 22. *Определим*

- $FIRST_k(\eta) = \{a_1 \dots a_j \,|\, (j < k \;\&\; \eta \to a_1 \dots a_j) \vee (j = k \;\&\; \eta \to a_1 \dots a_k\alpha)\} \cup \{\varepsilon \,|\, \eta \to^* \varepsilon\}$
- $FOLLOW_k(\eta) = \{a_1 \dots a_j \,|\, (j < k \;\&\; S \to^* \beta\eta a_1 \dots a_j) \vee (j = k \;\&\; S \to^* \beta\eta a_1 \dots a_k\alpha)\} \cup \{\$ \,|\, S \to^* \beta\eta\}$

Тогда G — LL(k)-грамматика $\Leftrightarrow \forall A \to \alpha, A \to \beta$ $(FIRST_k(\alpha\, FOLLOW_k(A)) \cap FIRST_k(\beta\, FOLLOW_k(A)) = \varnothing$

Определение 31.
- $FIRST_k(A) = \{a_1 \dots a_k \,|\, A \to a_1 \dots a_k\alpha\} \cup \{a_1 \dots a_j \,|\, j < k \;\&\; A \to a_1 \dots a_j\}$;
- *До исчерпания: $\forall A \to B_1 \dots B_n$, $FIRST_k(A) = FIRST_k(A) \cup first_k(FIRST_k(B_1) \dots FIRST_k(B_n))$, где $first_k$ — это k первых символов строки.*

Алгоритм задаёт фундированный порядок на множестве конфигураций $\mathrm{FIRST}_k(A_i)$, поэтому рано или поздно множества FIRST_k перестанут изменяться.

Добавляем правило из нового стартового символа $S_0 \to S\$$.

Определение 32.
- $FOLLOW_k(A) = \varnothing$ *для всех* $A \neq S$.
- *До исчерпания: $\forall B \to \beta$ и разбиений $\beta = \eta_1 A \eta_2$, $FOLLOW_k(A) = FOLLOW_k(A) \cup first_k(FIRST_k(\eta_2) FO$ где $first_k$ — это k первых символов строки.*

Алгоритм также задаёт wfo на множестве конфигураций $\mathrm{FOLLOW}_k(A_i)$.

Таблица $T_k(A, x)$ по нетерминалу A и lookahead-символам x вычисляет правило для парсинга.

Таблица LL(k)-разбора

for all $A \to \alpha$
  for all $x \in first_k(\mathrm{FIRST}_k(\alpha)\, \mathrm{FOLLOW}_k(A))$
    if $T_k(A, x)$ не задано, тогда $T_k(A, x) = A \to \alpha$
      else объявление о конфликте

PDA для LL(1)-грамматик

- Чтение терминалов с ленты происходит только в одном состоянии, при этом не меняется стек, но делается переход в другое состояние (учёт lookahead-ов).
- Во всех остальных состояниях только меняется стек. Переход ничего не читает с ленты и возвращает автомат в читающее состояние.

$S \to aBS$
$S \to \varepsilon$
$B \to aBB$
$B \to b$

$S \longrightarrow Z_0$ в PDA («дно»).

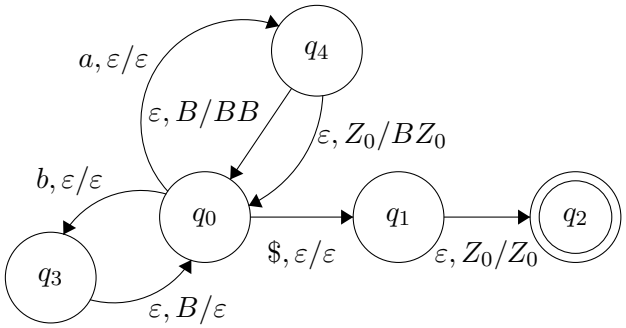


Рисунок 37 — К параграфу «PDA для LL(1)-грамматик»

Для LL(k) аналогично, но состояния будут соответствовать k последним прочитанным буквам.

Пример 10. *Стандартный алгоритм удаления ε-правил приводит к разрушению LL-свойств:*

$$\begin{array}{lcl} S & \to & ABC \\ A & \to & aA \,|\, d \\ B & \to & b \,|\, \varepsilon \\ C & \to & c \,|\, \varepsilon \end{array}$$

Утверждение (Куроки–Суонио)
Всякая LL(k)-грамматика может быть преобразована в LL(k+1)-грамматику без ε-правил.

Идея доказательства

Сначала избавимся от всех правил, начинающихся с nullable-нетерминала, путём введения notnull-двойников.

Затем присоединим все nullable-нетерминальные отрезки, стоящие в правых частях, к предшествующим им notnull-нетерминалам, и объявим полученные строки новыми нетерминалами.

В новых правилах nullable-кусок приписывается к самому последнему нетерминалу в правой части.

Пример устранения ε без разрушения LL-свойства

$$\begin{array}{ll} S \to ABC \,|\, Bk & A \to aA \,|\, d \\ B \to b \,|\, \varepsilon & C \to c \,|\, \varepsilon \end{array}$$

Сначала избавимся от обнуляемого нетерминала в начале правой части правила стандартным приёмом:

$$\begin{array}{lll} S \to ABC \,|\, B'k \,|\, k & A \to aA \,|\, d & \\ B \to b \,|\, \varepsilon & C \to c \,|\, \varepsilon & B' \to b \end{array}$$

Теперь присоединим правый обнуляемый контекст к A и протащим его по правилу переписывания для A:

$$S \to [ABC] \mid B'k \mid k \quad [ABC] \to a[ABC] \mid [dBC]$$
$$B \to b \mid \varepsilon \qquad C \to c \mid \varepsilon \quad B' \to b$$

Определяем правила переписывания для нетерминала $[dBC]$, соответствующие коллапсу всего обнуляемого контекста, его префиксов, и непустой развёртке префикса.

$$S \to [ABC] \mid B'k \mid k \quad [ABC] \to a[ABC] \mid [dBC]$$
$$[dBC] \to d[bC] \mid dc \mid d \quad B \to b \mid \varepsilon$$
$$C \to c \mid \varepsilon \qquad B' \to b$$

Аналогично обрабатываем нетерминал $[bC]$ и удаляем все правила для обнуляемых нетерминалов из грамматики.

$$S \to [ABC] \mid B'k \mid k \quad [ABC] \to a[ABC] \mid [dBC]$$
$$[dBC] \to d[bC] \mid dc \mid d \quad [bC] \to bc \mid b \quad B' \to b$$

Предложение 23. *Если G — $LL(k)$-грамматика, тогда вышеописанный алгоритм устранения ε-правил всегда завершается, поскольку ни на одном шаге не появляется новых нетерминалов, дважды присоединяющих в правый контекст один и тот же обнуляемый нетерминал грамматики G..*

Покажем, что в общем случае алгоритм может и не завершаться. Рассмотрим следующую грамматику, не являющуюся однозначной (и следовательно, $LL(k)$ ни для какого значения k).

$$S \to SS \mid a \mid \varepsilon$$

По алгоритму, устраним сначала обнуляемый правый нетерминал.

$$S \to S'S \mid S' \mid a \mid \varepsilon \quad S' \to S'S \mid S' \mid a$$

Теперь присоединим обнуляемые контексты (это оставшиеся вхождения S) и посмотрим, какие правила получатся для нового нетерминала $[S'S]$.

$$S \to [S'S] \mid S' \mid a \mid \varepsilon \quad S' \to [S'S] \mid S' \mid a \quad [S'S] \to S'SS \mid S'S \mid aS$$

Видно, что нужно порождать новый нетерминал, потому что обнуляемый контекст у правила $[S'S] \to S'SS$ — это уже два вхождения S. После его порождения опять получим правило вида $[S'SS] \to S'SSS$, и вообще, для каждого нового $[S'S^n]$ — правило $[S'S^n] \to S'S^{n+1}$. Значит, ни на какой итерации процесс не сходится.

Утверждения

- Ни одна леворекурсивная грамматика — не LL(k) ни для какого k.
- Всякая LL(k)-грамматика может быть преобразована в LL(k+1)-грамматику в GNF.

Определение 33. *CFL L — это $LL(k)$-язык, если для него существует $LL(k)$-грамматика.*

Существуют LL(k), но не LL(k-1)-языки.

LL(k)-языки

Рассмотрим язык $\{a^n(b^k d \mid b \mid c)^n\}$. Это LL(k)-язык:

$$\begin{array}{lcl} S & \to & aCA \\ C & \to & aCA \mid \varepsilon \\ A & \to & bB \mid c \\ B & \to & b^{k-1}d \mid \varepsilon \end{array}$$

Неформально: не LL(k-1) потому, что иначе бы имелась LL(k)-грамматика в GNF для L. В стеке разбора для префикса a^n оказалось бы больше, чем $2k-1$ символов. Подставляя варианты суффиксов к a^{n+k-1}, получаем противоречие (метод подмены).

Общая схема метода

- Рассматриваем сразу LL(k)-грамматику в GNF.
- Показываем, что в её стеке в состоянии α должно находиться не меньше, чем $f(k)$ разных символов при предъявлении тех же самых lookahead k символов.
- После данных k символов предъявляем длинный суффикс, отрезок которого должен контекстно-свободно выводиться из некоторого нетерминала в стеке.
- Предъявляем другой суффикс к тому же префиксу и lookahead, в котором не может быть такого отрезка, и подменяем вывод нетерминала.

Техника метода подмены

- Ищем такие два слова xfy, xfz, что $|f| = k$ и при чтении префикса x стек мог неограниченно разрастись (т.е. есть синхронная накачка хотя бы одной из пар x и y и x и z). Предполагаем высоту стека равной хотя бы $k + t$ (где t выбирается в зависимости от задачи).
- k символов в стеке при выталкивании точно распознают фрагмент f (lookahead). Рассматриваем, что может распознаться остальными t символами. Комбинируем распознанные фрагменты и показываем, что их комбинация не входит в язык.
- Либо если в стеке осталось t символов, а длина, например, y меньше t, тогда с этим стеком точно нельзя распознать y.

Пример подмены

Может ли язык $\{a^n b^n\} \cup \{a^n c^n\}$ задаваться LL(k)-грамматикой для некоторого k?

Пусть такое k нашлось (без ограничения общности — в GNF). Рассмотрим слово $a^{n+k}b^{n+k}$ и подберём такое n, что после чтения a^n LL-анализатор имеет в стеке не меньше $k + 2$ символов. Пусть последний символ — это Y. Он распознает некоторое подслово суффикса b^{n+k}, а именно b^s. Теперь подменим строку на $a^{n+k}c^{n+k}$. В этом случае Y должен распознать некоторое c^t. Но значит, $a^{n+k}b^{n+k-s}c^t \in L$, что невозможно. □

Опциональный Else

Рассмотрим GNF-грамматику для языка $\{a^n b^m \mid n \geq m\}$. Положим $w = a^{n+k}\, b^{n+k}$, где n таково, что в стеке на момент чтения a^n не меньше, чем $k + 2$ нетерминала. Тогда при подмене w на a^{n+k+1} из стека достанется только k нетерминалов, и слово a^{n+k+1} распознаться не сможет.

Следствие
Опциональный else не парсится с помощью LL-разбора.

«Плавающий else» делает грамматику неоднозначной. Договорённость по приоритетам (связывание с ближайшим if) снимает неоднозначность, но по префиксу выражения всё ещё невозможно установить, какой if имеется в виду: короткий или длинный.

Предложение 24.
- *LL(k)-языки не замкнуты относительно объединения.*
- *LL(k)-языки не замкнуты относительно пересечения с регулярным языком (пример: $\{a^n b a^n b\} \cup \{a^n c a^n c\}$ — не LL(k)-язык).*
- *Если L — LL(k)- нерегулярный язык, то его дополнение — не LL(k) ни для какого k.*
- *(Более сильное утверждение) Если $L_1 \cup L_2$ — регулярный язык, и L_1 — нерегулярный LL(k)-язык, то L_2 — не LL(k) ни для какого k.*