Дискретная математика

Контекстно-свободные языки

Константин Чухарев

Осень 2026

Грамматики

> *Бесцветные зелёные идеи яростно спят.*
>
> — ***Ноам Хомский***

КС-грамматика

Регулярные языки — простейшая ступень. КС-грамматики снимают ограничение.

Определение 1 (КС-грамматика (*context-free grammar*))

Четвёрка $G = (V, \Sigma, P, S)$:

- V — нетерминалы.
- Σ — терминалы.
- P — правила $A \to \alpha$, где $A \in V$ — один нетерминал, а $\alpha \in (V \cup \Sigma)^*$ — строка из терминалов и нетерминалов.
- S — стартовый нетерминал.

$$L(G) = \left\{ w \in \Sigma^* \;\middle|\; S \overset{*}{\to} w \right\}.$$

Порождение

Грамматика описывает язык порождением.

Правила конечны, слова — бесконечно много.

$a^n b^n$

Пример

$$S \to aSb \mid \varepsilon.$$

Вывод: $S \to aSb \to aaSbb \to aaaSbbb \to a^3b^3$.

Не регулярен, но контекстно-свободен.

На каждом шаге вывода слева появляется одна a, а справа — одна b.

Нужна память о числе — конечному автомату не под силу.

Язык Дика

Пример

Сбалансированные скобки:

$$S \to (S)S \mid \varepsilon.$$

Слово $(())()$ выводится так: раскрываем пары скобок и завершаем правилом $S \to \varepsilon$.

Вложенность — всюду:

- программы,
- выражения,
- разметка.

Распознать конечным автоматом нельзя.

Нужна память о глубине.

Грамматика предложений

Пример

Правила над словами:

$$S \to \text{NP VP},$$

$$\text{NP} \to \text{Det } N \mid N, \quad \text{VP} \to V \text{ NP} \mid V,$$

$$\text{Det} \to \text{старый}, \quad N \to \text{кот} \mid \text{мышь}, \quad V \to \text{ловит} \mid \text{бежит}.$$

Левый вывод: $S \to \text{NP VP} \to \text{Det } N \text{ VP} \to$ старый кот ловит мышь.

Слово «кот старый» не выводится: определение стоит только перед именем.

Язык — не все строки над алфавитом, а фразы со структурой.

Структуру задают правила, и она наследуется выводом.

Автоматы с магазинной памятью

PDA

Определение 2 (PDA (*pushdown automaton*))

Семёрка $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$:

- Q — состояния, Σ — входной алфавит.
- Γ — алфавит стека, $Z_0 \in \Gamma$ — донный символ.
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$.
- q_0 — начальное, $F \subseteq Q$ — принимающие.

Переход $(q', \gamma) \in \delta(q, a, A)$: из q, прочитав a, увидев A на вершине, перейти в q', заменив A строкой γ.

Стек — память о глубине вложенности.

Недетерминизм — как в НКА: существует путь.

Принятие слова

Определение 3 (Конфигурация и принятие)

Конфигурация — тройка (q, w, γ): состояние, остаток входа, стек.

- **По принимающему состоянию**: путь к (q, ε, γ) с $q \in F$.
- **По пустому стеку**: путь к $(q, \varepsilon, \varepsilon)$.

Оба критерия эквивалентны.

Распознавание $a^n b^n$

Пример

- Фаза q_0: на каждую a кладём A в стек.
- Первая b: фаза q_1, на каждую b снимаем A.
- Принятие по пустому стеку.

Прогон a a b b (дно — Z_0):

$$(q_0, aabb, Z_0) \to (q_0, abb, AZ_0) \to (q_0, bb, AAZ_0)$$
$$\to (q_1, b, AZ_0) \to (q_1, \varepsilon, Z_0) \to (q_1, \varepsilon, \varepsilon).$$

aba отвергается: в фазе q_1 символ a читать нельзя.

Глубина стека — сколько a ещё не нашли пару.

КСГ и PDA равносильны

Теорема 1 (КСГ и PDA)

Язык порождается КС-грамматикой тогда и только тогда, когда распознаётся МП-автоматом (PDA, автоматом с магазинной памятью).

Доказательство

От грамматики к автомату: автомат имитирует левый вывод, храня непрочитанную часть вывода в стеке. Верхний символ стека — терминал: снимаем его и сверяем с входной буквой. Верхний символ — нетерминал A: выбираем правило $A \to \beta$, снимаем A и кладём β. Автомат принимает, когда стек пуст и вход прочитан, и допускаемые им слова ровно те, что выводятся грамматикой.

КСГ и PDA: от автомата к грамматике

КСГ $\to$ PDA: автомат имитирует левый вывод.

PDA $\to$ КСГ: нетерминалы кодируют переходы со стеком.

Доказательство

От автомата к грамматике: кодируем состояния и стек нетерминалами вида $[pXq]$ — переместиться из p в q, удалив из стека X. Переход из p в r, снимающий X и кладущий $Y_1 \ldots Y_m$ при чтении a, даёт правило $[pXq] \to a[rY_1q_1][q_1Y_2q_2]\ldots[q_{m-1}Y_mq]$ со вспомогательными состояниями $q_1, \ldots, q_{m-1}$. Индукцией по числу переходов показывается, что такой нетерминал выводит строку тогда и только тогда, когда автомат проходит соответствующий путь.

Структура КС-языков

Дерево разбора

Определение 4 (Дерево разбора)

Корнем служит S, внутренние вершины помечены нетерминалами, а листья — терминалами.

Дети вершины A слева направо — правая часть применённого правила $A \to \alpha$.

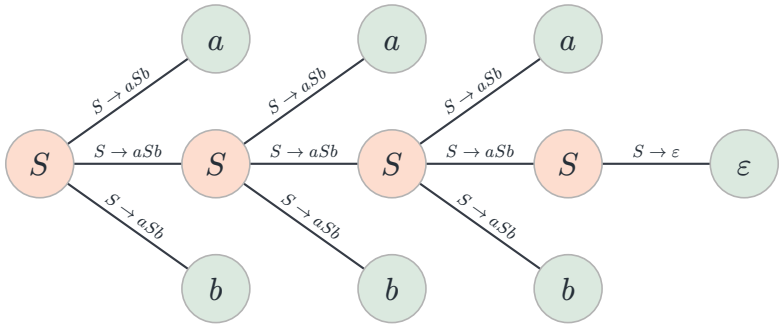
Метки листьев слева направо дают слово w.

Пример

Слово a a b b из $S \to aSb \mid \varepsilon$: корень S раскрывается в aSb, внутренний S — снова в aSb, самый внутренний — в ε.

Дерево показывает: первая a спарена с последней b, вторая — с предпоследней.

Дерево: a^3b^3



Дерево растёт слева направо: корень S слева, листья — справа.

На каждой стрелке — применённое правило: три раскрытия $S \to aSb$ и завершение $S \to \varepsilon$.

Пустой лист ε символов не даёт: листья $a, a, a, \varepsilon, b, b, b$ складываются в слово a^3b^3.

Видимая вложенность

Строка прячет, какая скобка закрывает какую.

Дерево разбора делает вложенность видимой.

Неоднозначность

Определение 5 (Неоднозначная грамматика)

Грамматика неоднозначна, если некоторое слово языка имеет более одного дерева разбора.

Два дерева — два смысла одной строки.

Неоднозначность приоритетов устраняют уровнями правил: слагаемые собираются из множителей.

Пример: два дерева

Пример

$$E \to E + E \mid E * E \mid (E) \mid \mathrm{id}.$$

Слово $\mathrm{id} + \mathrm{id} * \mathrm{id}$ — два дерева:

- Склеить сначала $\mathrm{id} + \mathrm{id}$: чтение $(\mathrm{id} + \mathrm{id}) * \mathrm{id}$.
- Склеить сначала $\mathrm{id} * \mathrm{id}$: чтение $\mathrm{id} + (\mathrm{id} * \mathrm{id})$.

При числах значения расходятся: $(2 + 3) \cdot 4 = 20$, но $2 + (3 \cdot 4) = 14$.

Лемма о накачке

Теорема 2 (Лемма о накачке для КС-языков (*pumping lemma*))

Если L контекстно-свободен, то есть $p \geq 1$: любое $w \in L$ длины $|w| \geq p$ разбивается как $w = uvxyz$:

- $|vy| > 0$;
- $|vxy| \leq p$;
- $uv^i xy^i z \in L$ для всех $i \geq 0$.

Накачиваются две подстроки одновременно и симметрично.

Регулярный случай — одна подстрока (цикл). Здесь две (стек растёт и убывает).

$a^n b^n c^n$ не КС

Пример

Возьмём $w = a^p b^p c^p$.

Подстрока vxy длины $\leq p$ не захватывает и a, и c: между ними стоят все b.

Так как vxy не содержит одновременно a и c, при $i = 0$ или $i = 2$ одна из букв перестаёт накачиваться, и равенство количеств нарушается.

Значит, $a^n b^n c^n$ не контекстно-свободен.

Один стек — один счётчик.

Два независимых счётчика уводят за пределы класса.

Разбор

Обратная задача

КС-грамматика задаёт язык порождением. Разбор — обратная задача: по строке найти её вывод.

Определение 6 (Разбор)

Дана строка w и грамматика G. Определить, лежит ли w в $L(G)$, и если да — построить дерево разбора.

Именно это делает компилятор: текст программы превращается в синтаксическое дерево.

Пример

Для $S \to aSb \mid \varepsilon$ и строки aabb: $S \to aSb \to aaSbb \to aabb$ — вывод, он же путь в дереве разбора.

Направления разбора

- Нисходящий разбор начинает со стартового символа и раскрывает продукции.
- Восходящий разбор сворачивает правые части продукций к стартовому символу.

Трудность — выбор: какую продукцию применить, где и что свернуть.

LL и LR

Определение 7 (Классы LL(k) и LR(k))

- Грамматика LL(k) разбирается нисходяще с подглядыванием на k символов вперёд.
- Грамматика LR(k) разбирается восходяще.
- Большинство грамматик языков программирования — LR (или их упрощение LALR).

Для LR-грамматик существуют генераторы парсеров:

- yacc,
- bison,
- ANTLR.

LL- и LR-языки — собственные подклассы КС-языков.

Недетерминизм — цена выразительности: не всякая КС-грамматика разбирается детерминированно.

Рекурсивный спуск

Определение 8 (Рекурсивный спуск)

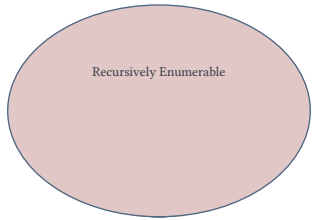
Простейший нисходящий парсер состоит из одной функции на каждый нетерминал и пишется вручную.

Функция для A пробует правые части правил $A \to \alpha$ по очереди.

Так устроены парсеры компиляторов, написанные вручную.

Иерархия Хомского

Ступени



- **Регулярные**: конечные автоматы.
- **Контекстно-свободные**: PDA.
- **Контекстно-зависимые**: линейно-ограниченные автоматы.
- **Рекурсивно перечислимые**: машины Тьюринга.

Каждая ступень строго больше: $a^n b^n$ — КС, но не регулярный.

Разрешимость

Подъём по иерархии оплачивается потерей алгоритмических гарантий.

- Для регулярных языков разрешимы все стандартные вопросы: эквивалентность, пустота, включение.
- Для КС-языков разрешима пустота, но эквивалентность — уже нет.
- Для рекурсивно перечислимых неразрешима даже пустота (следующая лекция).

Контекстно-зависимые

Определение 9 (КЗ-грамматика (*context-sensitive grammar*))

Правила вида $\alpha A \beta \to \alpha \gamma \beta$: нетерминал A заменяется на непустую γ, только если вокруг стоят α и β.

Замена чувствительна к контексту — отсюда название.

Пример

Грамматика для $a^n b^n c^n$:

$$S \to aSBC \mid aBC, \quad CB \to BC,$$
$$aB \to ab, \quad bB \to bb, \quad bC \to bc, \quad cC \to cc.$$

- Правило $CB \to BC$ переставляет соседние B и C.
- Остальные превращают их в b и c.

Монотонность

КЗ-правило не укорачивает строку: правая часть длиннее левой на $|\gamma| - 1 \geq 0$ символов.

В выводе слова w ни одна промежуточная строка не длиннее $|w|$. Строк подходящей длины над конечным алфавитом — конечное число.

Распознавание перебором заведомо завершается: кандидатов конечное число.

Монотонные грамматики (все правила $\alpha \to \beta$ с $|\beta| \geq |\alpha|$) порождают в точности контекстно-зависимые языки.

Линейно-ограниченный автомат

Стек доступен только с вершины, а контекстно-зависимым языкам нужна память с произвольным доступом.

Определение 10 (Линейно-ограниченный автомат)

Линейно-ограниченный автомат — машина Тьюринга, длина ленты которой ограничена линейной функцией от длины входа.

Распознаёт в точности контекстно-зависимые языки.

Пример

ЛОА распознаёт $a^n b^n c^n$:

1. Стирает крайние a и c попарно.
2. Сверяет оставшиеся b, умещаясь на ленте длины $O(n)$.

Мост к машине Тьюринга

Мост к следующей лекции: полная машина Тьюринга — та же лента без ограничения длины.

КЗ-языки разрешимы, но распознавание дороже: перебор растёт экспоненциально с длиной слова.

КС-языки на практике

- КС-грамматики — синтаксис языков программирования.
- Парсеры строят деревья разбора.
- Язык Дика — вложенность скобок в коде.

Нотация Бэкуса–Наура (BNF) — синтаксическая запись КС-грамматик.

Парсеры — алгоритмы построения дерева разбора.

Иерархия Хомского — карта выразительности.

От регулярных к рекурсивно перечислимым.

Что мы поняли?

- КС-грамматика описывает язык порождением: конечные правила дают бесконечно много слов.
- Стек — память о глубине вложенности. КС-языки — ровно те, что распознаются МП-автоматом (PDA).
- Дерево разбора делает вложенность видимой. Два дерева у одного слова — два смысла.
- Один стек — один счётчик: $\{a^n b^n c^n\}$ требует двух независимых счётчиков и выходит за пределы КС.

Вопросы для самопроверки

- Определите КС-грамматику.
- Запишите грамматику для $\{a^n b^n\}$.
- Что такое язык Дика?
- Почему $a^n b^n$ распознаётся PDA, но не ДКА?
- Сформулируйте эквивалентность КСГ и PDA.
- Когда грамматика неоднозначна?
- Сформулируйте лемму о накачке для КС-языков.
- Почему $\{a^n b^n c^n\}$ не контекстно-свободен?
- Опишите иерархию Хомского.