

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

Теория формальных языков
2023 г.



Алфавитные префиксные грамматики

Определение APG

Дана SRS \mathcal{S} с правилами переписывания двух видов:

$$a_i \rightarrow b_1 \dots b_n \quad a_i \rightarrow \varepsilon$$

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.

Утверждение

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



Алфавитные префиксные грамматики

Утверждение

Язык $L\langle S, w_0 \rangle$ регулярен.

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

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

- 1 Пусть $a \rightarrow b_1 \dots b_n$ и $\exists b_i (\neg(b_i \twoheadrightarrow \varepsilon) \ \& \ \forall j (j < i \Rightarrow b_j \twoheadrightarrow \varepsilon))$. Тогда добавим в G правила $A \rightarrow B_1 b_2 \dots b_n, A \rightarrow B_2 b_3 \dots b_n, \dots, A \rightarrow B_i b_{i+1} \dots b_n, A \rightarrow a$.
- 2 Если такого b_i нет, добавляем в G все правила вида $A \rightarrow B_1 b_2 \dots b_n, \dots, A \rightarrow B_{n-1} b_n, A \rightarrow B_n, A \rightarrow a$.
- 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 $S \rightarrow \varepsilon$.



Алфавитные префиксные грамматики

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

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

- 1 Пусть $a \rightarrow b_1 \dots b_n$ и $\exists b_i (\neg(b_i \twoheadrightarrow \varepsilon) \ \& \ \forall j (j < i \Rightarrow b_j \twoheadrightarrow \varepsilon))$. Тогда добавим в G правила $A \rightarrow B_1 b_2 \dots b_n, A \rightarrow B_2 b_3 \dots b_n, \dots, A \rightarrow B_i b_{i+1} \dots b_n, A \rightarrow a$.
- 2 Если такого b_i нет, добавляем в G все правила вида $A \rightarrow B_1 b_2 \dots b_n, \dots, A \rightarrow B_{n-1} b_n, A \rightarrow B_n, A \rightarrow a$.
- 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 $S \rightarrow \varepsilon$.

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



Неалфавитные грамматики

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

Языки префиксных грамматик регулярны.

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



От ДКА к префиксной грамматике

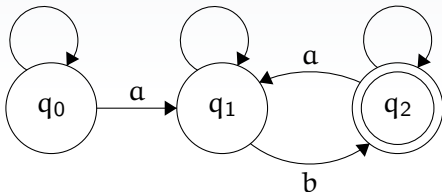
В данном алгоритме рассматривается минимальный ДКА для языка.

- 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: $w_i \rightarrow w_k a_j$.
- Начальными словами объявим слова из классов эквивалентности, лежащих в языке автомата.



От ДКА к префиксной грамматике

Построим префиксную грамматику для языка автомата:



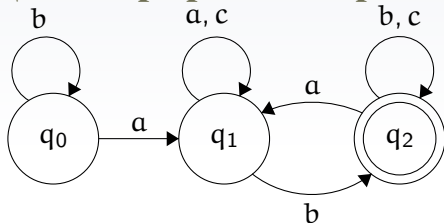
Для q_0 ведущим словом будет b , для q_1 — a , для q_2 ведущее ab (оно же стартовое слово).

Правила префиксной грамматики:

- $b \rightarrow bb$ (в q_0 входит лишь одна стрелка)
- $a \rightarrow aa$ (стрелки из q_1 в себя)
- $a \rightarrow ba$ (стрелка из q_0 в q_1)
- $ab \rightarrow ab$ (стрелка из q_1 в q_2)
- $ab \rightarrow abc$ $ab \rightarrow abb$ (стрелки из q_2 в себя)



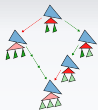
От ДКА к префиксной грамматике



Стартовое слово: ab . Правила переписывания:

- $b \rightarrow bb$ (в q_0 входит лишь одна стрелка)
- $a \rightarrow aa$ $a \rightarrow ac$ (стрелки из q_1 в себя)
- $a \rightarrow ba$ (стрелка из q_0 в q_1)
- $ab \rightarrow ab$ (стрелка из q_1 в q_2)
- $ab \rightarrow abc$ $ab \rightarrow abb$ (стрелки из q_2 в себя)

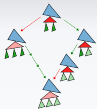
Результат похож на обращенные правила трансформационного моноида, но учитывает префиксность: нет смысла переписывать $c \rightarrow cc$, если c может встретиться только после буквы a , либо после префикса ab .



Ограничения регулярных грамматик

- (синтаксический моноид) Слова лишь конечно различимы относительно правил переписывания
- (префиксные грамматики) Доступ лишь к началу (концу) слова
- (накачки) Все циклы в автоматах асинхронны

Что будет, если снять эти условия?



Ограничения регулярных грамматик

- (синтаксический моноид) Слова лишь конечно различимы относительно правил переписывания
- (префиксные грамматики) Доступ лишь к началу (концу) слова
- (накачки) Все циклы в автоматах асинхронны

Что будет, если снять эти условия?

Структура вывода — дерево, а не последовательность.

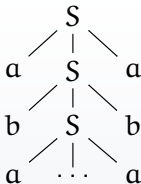
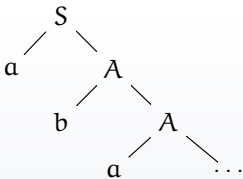


Ограничения регулярных грамматик

- (синтаксический моноид) Слова лишь конечно различимы относительно правил переписывания
- (префиксные грамматики) Доступ лишь к началу (концу) слова
- (накачки) Все циклы в автоматах асинхронны

Что будет, если снять эти условия?

Структура вывода — дерево, а не последовательность.





Контекстно-свободные грамматики

Определение

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

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



Контекстно-свободные грамматики

Определение

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

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

Грамматика G_1

S	\rightarrow	SS
S	\rightarrow	(S)
S	\rightarrow	ε

Грамматика G_2

S	\rightarrow	B	R	\rightarrow	$)$
B	\rightarrow	$(RB$	R	\rightarrow	$(RR$
B	\rightarrow	ε			



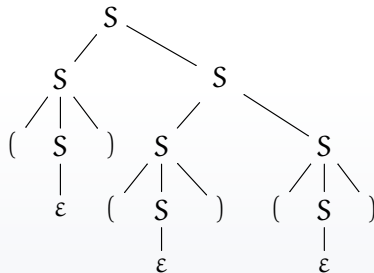
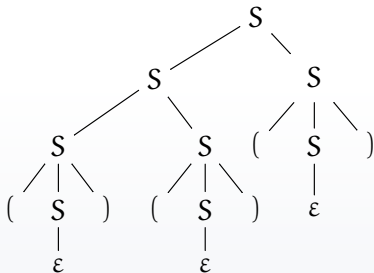
Неоднозначность разбора

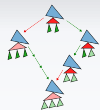
Грамматика G_1 для языка Дика

$S \rightarrow SS$

$S \rightarrow (S)$

$S \rightarrow \epsilon$





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 $A \rightarrow \beta$. Левосторонний разбор S — разбор, каждый шаг которого левосторонний.

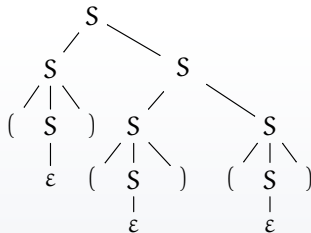
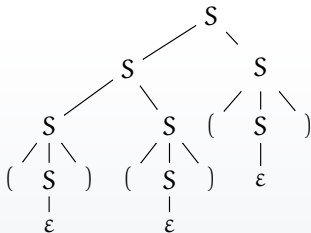


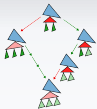
Левосторонний разбор

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 $A \rightarrow \beta$. Левосторонний разбор S — разбор, каждый шаг которого левосторонний.

Левосторонний разбор не обязательно единственный, см. ниже.

$$S \rightarrow SS \quad S \rightarrow (S) \quad S \rightarrow \varepsilon$$





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 $A \rightarrow \beta$. Левосторонний разбор S — разбор, каждый шаг которого левосторонний.

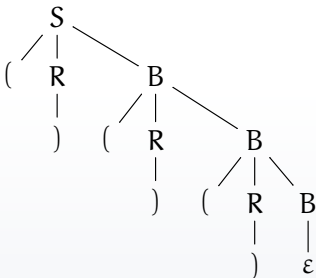
Утверждение

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



(Не)однозначность грамматик

Грамматика G_2 для языка Дика

$$\begin{array}{ll} S \rightarrow B & R \rightarrow) \\ B \rightarrow (RB & R \rightarrow (RR \\ B \rightarrow \varepsilon & \end{array}$$


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



Устранение ε -правил

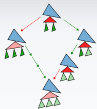
$A \in N$ коллапсирует, если $A \rightarrow \varepsilon \in P$ или $A \rightarrow \alpha \in P$ и все элементы α коллапсируют.



Устранение ε -правил

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

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



Устранение ε -правил

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

- Если $\varepsilon \in L(G)$, тогда добавляем новый стартовый символ S_0 и правила $S_0 \rightarrow \varepsilon$, $S_0 \rightarrow S$.
- Стираем все правила $B_i \rightarrow \varepsilon$, кроме $S_0 \rightarrow \varepsilon$.
- Для всех правил $A \rightarrow \alpha_1 B_i \alpha_2$, где $B_i \in \text{Nullable}$, добавляем правила $A \rightarrow \alpha_1 \alpha_2$.



Устранение ε -правил

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

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



Устранение ε -правил

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

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



Уничтожение цепных правил

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



Нормальная форма Хомского

Определение

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



Нормальная форма Хомского

Определение

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

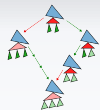
- Устраняем ε -правила.
- Устраняем цепные правила.
- $\forall a \in \Sigma$ таких, что a входит в правую часть правила, отличную от a , заводим нетерминал-охранник G_a , строим правило $G_a \rightarrow a$, и во всех правых частях, кроме совпадающих с a , заменяем a на G_a .
- $\forall A \rightarrow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



Смысл нормальной формы Хомского

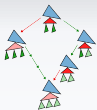
- 1 Неукорачивающие применения правил
- 2 Нет пустых переходов — правила либо финальные, либо удлиняющие
- 3 Контролируемый рост длины сентенциальной формы от количества шагов разбора

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



Недостижимость и заикливание

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



Недостижимость и заикливание

- Стартовый нетерминал $S \in N$ достижим.
 - Нетерминал $A \in N$ достижим, если существует правило $B \rightarrow \alpha$ такое, что $|\alpha|_A \geq 1$ и B достижим.
-
- Если существует правило $A \rightarrow w$, $w \in \Sigma^*$, A порождающий.
 - Если $A \rightarrow \alpha$ и $\forall B_i (|\alpha|_{B_i} \geq 1 \Rightarrow B_i \text{ порождающий})$, то A порождающий.
-
- 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.



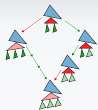
Проверка корректности рекурсивных алгоритмов

- 1 Завершаемость — фундированность — искомое множество M нетерминалов не может уменьшаться, и количество нетерминалов грамматики конечно.



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 $w_1(w_2)^nw_3$, причём w_1w_3 также входит в язык сентенциальных форм G .

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

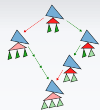
Но ещё есть шаг порождения w_2 , также выбрасывающий последовательность терминалов.



Лемма о накачке КС-языков

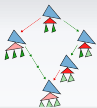
Лемма о накачке (разрастании)

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 $L(G)$.



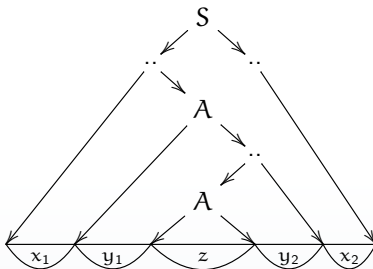
Лемма о накачке КС-языков

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 .



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 .



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





Пример применения

Парсинг в Python

Проанализировать язык

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



Пример применения

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:

$\forall \exists \forall \exists \exists$. 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.



Теоретико-игровая интерпретация

- 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).



Замыкания

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



Техника применения

Сужение перебора

Если в язык L входят под слова произвольной формы из Σ^+ , где $|\Sigma| > 1$, тогда, скорее всего, потребуется пересечь L с регулярным языком, чтобы облегчить поиск свидетельства о ненакачиваемости. Пример: язык $\{w_1 w_1 w_2 \mid |w_1|_a = |w_2|_a\}$. Пересечение этого языка с $ba^*ba^*ba^*$ 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 .



Техника применения

Работа с отрицанием

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 $L = \{w \mid w \neq a^{n^2} \text{ \& } w \in \{a, b\}^*\}$ не является КС. Для начала заметим, что слова L содержат произвольные под слова 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.

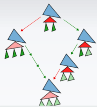
- Антагонист выбирает p .
- Наша задача — подобрать такое k , что $\forall p' \exists i, m(p' < p \Rightarrow k + p' * i = m^2)$. То есть включить возможность взятия любого такого p' в наше значение k как конструктивного элемента для построения квадрата числа.
- Возьмём $k = (p!)^2 + 1$. Тогда при любом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



Ещё пример применения

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 $ba^i b$, а весь язык — вид $L' = \{ba^n bba^n ba^n\}$.

- Абеляр выбирает p . Элоиза строит слово $ba^p b^2 a^p ba^p$. Абеляру предоставляется возможность построить его разбиение на $x_1 y_1 z y_2 x_2$.
- 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.



Ещё пример применения

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 $L' = \{ba^n bba^n ba^n\}$.

- Абеляр выбирает p . Элоиза строит слово $ba^p b^2 a^p ba^p$. Абеляру предоставляется возможность построить его разбиение на $x_1 y_1 z y_2 x_2$.
- 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.
 - $ba^p b^2 a^{p+i*k}ba^{p+j*k}$ — теряется свойство палиндрома.
 - $ba^{p+(i+j)*k}b^2a^pba^p$ — теряется свойство палиндрома, при накачке только второго подслова a^p аналогично.
 - $ba^p b^2 a^p ba^{p+(i+j)*k}$ — некорректный подсчёт букв a в w .



Языки, накачиваемые обманно

Некоторые не КС-языки тоже накачиваются, например,
 $\{a^m b^n c^n d^n \mid m > 0\} \cup \{b^i c^j d^k\}$.



Языки, накачиваемые обманно

Некоторые не КС-языки тоже накачиваются, например,
 $\{a^m b^n c^n d^n \mid m > 0\} \cup \{b^i c^j d^k\}$.

Действительно, если слово языка содержит буквы a , тогда мы можем взять $y_1 y_2 = a^i$. Иначе накачку можно выбрать произвольно.



Языки, накачиваемые обманно

Некоторые не КС-языки тоже накачиваются, например, $\{a^m b^n c^n d^n \mid m > 0\} \cup \{b^i c^j d^k\}$.

Действительно, если слово языка содержит буквы a , тогда мы можем взять $y_1 y_2 = a^i$. 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.



Языки, накачиваемые обманно

Некоторые не КС-языки тоже накачиваются, например, $\{a^m b^n c^n d^n \mid m > 0\} \cup \{b^i c^j d^k\}$.

Действительно, если слово языка содержит буквы a , тогда мы можем взять $y_1 y_2 = a^i$. 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.

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 .