

- При исследовании накачек не забываем про отрицательную накачку (0 итераций накачки).
- Если язык подозрителен на не-CFL, пытаемся свести задачу к известным с помощью пересечения с регулярными языками и гомоморфизмов.
- Если язык подозрителен на CFL, пытаемся разбить его на непересекающиеся более простые языки.
- За отсутствие состояния-ловушки в минимальных автоматах штрафа не будет.



Типы языков

- Язык — пересечение. Аккуратно строим два элементарных языка и дальше действуем автоматически.

Язык слов, которые одновременно являются правильно построенными логическими формулами и реверсами правильно построенных логических формул. Алфавит $\{T, F, \vee, \neg, \&, (,)\}$.



Типы языков

- Инфиксы, суффиксы etc известного языка. Пишем грамматику исходного языка в форме Грейбах и на её основе грамматику, которая пропускает порождение префиксных или суффиксных символов.

Язык всевозможных префиксов регулярного выражения только с * и конкатенацией, не итерирующего пустое слово. Алфавит $\{a, *, (,)\}$.

Грамматика:
$$\begin{array}{ll} S \rightarrow (S)^* S & S \rightarrow (S)^* \\ S \rightarrow a S & S \rightarrow a \end{array}$$

Грамматика префиксов:

$$\begin{array}{lll} S \rightarrow (S)^* S & S \rightarrow (S)^* & S \rightarrow (S) \\ S \rightarrow (S & S \rightarrow a S & S \rightarrow a \\ S \rightarrow (& & \end{array}$$



Типы языков

- Вычисляющий язык (вычисляет константу). Пытаемся понять, какие свойства подвыражений порождают требуемое значение.

Язык КС-грамматик, порождающих пустое слово. Алфавит $\{S, A, B, \varepsilon, \rightarrow, \alpha, \$\}$ ($\$$ — разделитель).

Выполняется, если:

- Есть правило $S \rightarrow \varepsilon$.
- Есть правило вида $S \rightarrow [\text{строка только из нетерминалов}]$
- Данные нетерминалы можно разделить на классы (их, самое большее, будет 3) такие, что через n промежуточных нетерминалов нетерминал переписывается в ε .



Типы языков

- Языки путей в грамматике. Пытаемся посмотреть через линзу гомоморфизма и пытаться свести к языкам, порождаемым этой грамматикой.
- Языки с условиями вида $|w|_{v_1} = |w|_{v_2}$. Пытаемся понять, какая взаимосвязь есть между вхождениями v_1 и v_2 . Если взаимосвязь исчерпывается конечным числом случаев — язык регулярный, иначе строим КС-язык, но имеем в виду перекрытия!



Определение типа языка

Подсказки, что язык — не контекстно-свободен:

- Если косвенно встречается указание на равенство подструктур (неограниченных индексов в натуральных числах и т.д.)
- Если не удаётся разбить слово на фрагменты, соответствующие ПСП, имеющие синхронные «накачки».

Фрагменты должны чётко отделяться друг от друга! Иначе могут возникнуть иллюзии: см. $\{v_1 w v_2 w \mid w \in \{a, b\}^+\}$ (и сравним с $\{v_1 w v_2 w \mid w \in a, b^+ \ \& \ v_2 \in b^+\}$).

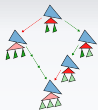


Определение типа языка

Подсказки, что язык — не регулярен:

- Если косвенно встречается указание на подсчёт вхождений, причём в промежуточном состоянии подсчёта могут быть сколь угодно большие значения разницы числа вхождений.
- Если есть намёк на ПСП.

То же замечание, что и для не КС (ищем чётко различающий инфикс, а если есть сомнения — пересекаем с регулярным языком).



Структура тестов на контесте

- 1 регулярно-замкнутые ($20 * 0, 5 + 3$);
- 2 линейные КС с накачкой ($5 * 1 + 10 * 2$);
- 3 регулярные с поглощением накачек ($10 * 2 + 5 * 3 + 1$);
- 4 с произвольно длинными фрагментами накачек ($10 * 2 + 5 * 3$);
- 5 КС, содержащие в себе линейный КС с накачкой ($5 * 3 + 2$);
- 6 синтаксические ошибки ($5 * 0, 5$);
- 7 «монстры» (КС-грамматики со сложно разрешимой регулярностью) (5 примеров).



Регулярно-замкнутые грамматики

Если в языке нет ни одного нетерминала, который переписывается так: $N_i \rightarrow^* \alpha N_i \beta$, где $\alpha \neq \varepsilon$ и $\beta \neq \varepsilon$, то язык регулярен.

Доказательство: можно скачать конструкцию со статьи Mohri–Nederhof (см. non-self-embedding grammars)

- Выделяем нетерминалы, переписывающиеся только в ε , и стираем их. Они не влияют на язык.
- Строим множества взаимно-рекурсивных нетерминалов:

$$M_s = \{T \mid \forall T_i, T_j \in M_s (T_i \rightarrow^* \alpha T_j \beta \ \& \ T_j \rightarrow^* \gamma T_i \delta)\}.$$

Если хотя бы одно взаимно-рекурсивное замыкание имеет вид $T_i \rightarrow^* \alpha \gamma T_i \delta \beta$, причём языки обеих сентенциальных форм $\alpha \gamma$, $\delta \beta$ отличны от языка $\{\varepsilon\}$ (и не пусты), тогда грамматика не регулярно-замкнута.



Линейные КС с накачкой

Если $S \rightarrow^* \alpha S \beta$ и $S \rightarrow^* \gamma$, причём существуют слова w_α , w_β , w_γ такие, что $w_\tau \in L(\tau^*)$ & $\forall \tau_1 (\tau_1 \neq \tau \Rightarrow w_\tau \notin L(\tau_1^*))$, то язык — не регулярен.

Доказательство: пересечём с регуляркой, вынуждающей итерироваться $(x_1 w_\alpha x_2)^* (x_3 w_\gamma x_4) (x_5 w_\beta x_6)^*$, и попробуем накачать фрагмент, захватывающий целиком w_β .

Здесь языки линейные (т.е. ровно с 1 нетерминалом в правой части), но может быть несколько вложенных подвыводов (с чёткой границей саморекурсивного фрагмента слева и справа).



Линейные КС с накачкой

Если $S \rightarrow^* \alpha S \beta$ и $S \rightarrow^* \gamma$, причём существуют слова w_α , w_β , w_γ такие, что $w_\tau \in L(\tau^*)$ & $\forall \tau_1 (\tau_1 \neq \tau \Rightarrow w_\tau \notin L(\tau_1^*))$, то язык — не регулярен.

Доказательство: пересечём с регуляркой, вынуждающей итерироваться $(x_1 w_\alpha x_2)^* (x_3 w_\gamma x_4) (x_5 w_\beta x_6)^*$, и попробуем накачать фрагмент, захватывающий целиком w_β .

Достаточно, чтобы нашлось w_γ , не входящее ни в один язык (α^* и β^*). Это рассуждение работает надёжно, только если языки α , β не допускают пустое слово (т.е., например, если $\alpha = (a|b)^*$, β — любой регулярный, а $\gamma = c$, то весь язык всё равно регулярен, см. идемпотентность языков).



Регулярные с поглощением накачек

Если $T \rightarrow \alpha T \beta$, где α, β — регулярные языки, причём $\alpha^* \approx \beta^*$, т.е. $\forall w (\exists i, k_1 (w^{k_1} \in \alpha^i) \Leftrightarrow \exists j, k_2 (w^{k_2} \in \beta^j))$, и к тому же для языка $T \rightarrow^* \gamma$ слов, не проходящих через этот рекурсивный цикл, также выполнено $\gamma^* \approx \beta^*$, то язык, описываемый T , регулярен.

Если $\alpha = \alpha\alpha$ или $\beta = \beta\beta$ (регулярные языки идемпотентны), то язык сразу регулярен, если γ регулярен.

Если нашлось w , опровергающее условие аппроксимации, то это возможная подсказка, с чем нужно пересекать грамматику для попытки поиска накачек. Если нашлись такие w и для α , и для β — это очень хорошая подсказка. Деталь: регулярные языки могут собираться по разным ветвям вывода, например $S \rightarrow a S b \mid a S a \mid a \mid b \mid b S a \mid b S b$. Имеем: $(a|b)^* = (a|b)^*$.



Звёздочки

Даже в линейном КС-языке регулярки α и β могут содержать звёздочки. А именно, если есть саморекурсия вида $T_i \rightarrow \xi T_i$, то ξ^* может появиться где угодно между строками w_i такими, что существует саморекурсия $T_i \rightarrow w_i T_i v_i$. Аналогично справа от T_i (в языке β).



С произвольно длинными фрагментами...

Если коммутативный образ языка L вынуждает соотношение $|w|_b < \sum k_j \cdot |w|_{a_j} + C_0$ для некоторых букв b, a_j , но при этом можно построить слово в L , содержащее произвольно длинный фрагмент v , такой что $|v|_b > \sum k_j \cdot |v|_{a_j} + C_0$, тогда язык L не регулярен.

Пример: $S \rightarrow a S a \mid b S a \mid b$ (смотрим на подслово b^n в префиксе).

Коммутативный образ многократно саморекурсивных правил строится перемещением саморекурсивных вызовов вслед за базисными. Положим $S \rightarrow b S S S \mid a$. Имеем $S \rightarrow (b S S)^+ S \mid a$, теперь проталкиваем вперёд все базисные случаи: $S \rightarrow (baa)^+ S \mid a$, и по лемме Ардена: $S = (baa)^+ a$.



С произвольно длинными фрагментами...

Если коммутативный образ языка L вынуждает соотношение $|w|_b < \sum k_j \cdot |w|_{a_j} + C_0$ для некоторых букв b, a_j , но при этом можно построить слово в L , содержащее произвольно длинный фрагмент v , такой что $|v|_b > \sum k_j \cdot |v|_{a_j} + C_0$, тогда язык L не регулярен.

Пример: $S \rightarrow a S a \mid b S a \mid b$ (смотрим на подслово b^n в префиксе).

Однородные фрагменты имеет смысл искать по последней либо первой развёртке T_i в саморекурсивном выводе. Если есть саморекурсия $T_i \rightarrow \alpha T_i \Phi$, где α — константа, то есть произвольно длинные фрагменты α^+ . Аналогично для $T_i \rightarrow \Psi T_i \beta$, где β — константа.



Синтаксис

Начальный нетерминал: $[S]$, пустое слово — $_$.

```
 $\langle \text{grammar} \rangle ::= \langle \text{rule} \rangle^+$   
 $\langle \text{rule} \rangle ::= \langle \text{nterm} \rangle \rightarrow \langle \text{term} \rangle^+$   
 $\langle \text{term} \rangle ::= \langle \text{nterm} \rangle \mid [a-z] \mid \_$   
 $\langle \text{nterm} \rangle ::= [[A-z]^+ [0-9]^*]$ 
```

- Всего из 80 тестов 5 будет с некорректным синтаксисом, и уже на этих пяти тестах можно с гарантией заработать 2,5 балла.
- Использование табуляций, лишних пробелов, а также различных вариантов перевода строки, и наличие лишних пустых строк синтаксическими ошибками не считаются.



Свидетельства

- Ответ: regular, non-regular, unknown — записывается в файл result.txt в корневой папке. Т.е. файл result.txt содержит ответ строго на один тест. По умолчанию не требуется сообщать, каким образом была установлена регулярность или нерегулярность.
- Если по ключу запуска будет возможно получить свидетельство ответа (свойство грамматики, которое дало возможность сделать вывод о её регулярности), то это добавит +3 балла к базовой сумме баллов (до повышающего множителя).
- Свидетельство ответа — в свободной форме. Записывается в лог, не являющийся файлом result.txt.



Проверка корректности

- Если просмотр кода после РК выявит грубую подгонку под тесты / рандомизацию, с вас снимаются все баллы по срезам (соответственно, награждаются следующие по очереди) и накладывается командный штраф 5 баллов за каждое нарушение.
- Если ваши же собственные секретные тесты окажутся неустойчивыми при контрольной проверке, ваши баллы за секретные тесты умножатся на 0,5.
- Недетерминированными алгоритмами пользоваться можно в том случае, если они выбирают путь разбора случаев. При этом могут дополнительно получаться результаты `unknown` или `timeout`, но не может получаться на одном прогоне `regular`, а на другом `non-regular` — это считается априори рандомизацией и штрафуются (см. п.1).