Вариант 6, 7, 8

- Найти (в пределе) регулярные язык префиксов и суффиксов, содержащие хотя бы одну итерацию.
- Для каждой пары «накачек» из выбранных регулярных префиксов и суффиксов проверить, что с достаточно высокой вероятностью слова из накачек формируют слово из языка $\mathscr{L}$.

Вариант 0, 1, 3, 9

- Дано разбиение $\langle \omega_1, \omega_2, \omega_3, \omega_4, \omega_5 \rangle$, предположительно описывающее серию слов $\omega_1 \omega_2^n \omega_3 \omega_4^n \omega_5$ в языке $\mathscr{L}$.
- Найти (в пределе) языки отдельно $\mathscr{L}(\omega_1)$, $\mathscr{L}(\omega_3)$, $\mathscr{L}(\omega_5)$.
- Проверить, что пары слов из $\mathscr{L}(\omega_1)$ и $\mathscr{L}(\omega_5)$ действительно совместны относительно накачек ω_2 и ω_4 в языке $\mathscr{L}$.

Вариант 2, 4, 5

- Найти языки разбиений относительно букв алфавита Σ слов языка $\mathscr{L}$.
- В рамках этих языков найти неразличимые элементы лексем и последовательные элементы лексем.

Варианты алгоритмов вывода

- (Чётная последняя цифра зачётки) алгоритм `L*`.
- (Нечётная последняя цифра зачётки) алгоритм `NL*`.

Если вы в числе приоритетных (оба сдали ЛР до 16 октября), то можете выбрать в этой позиции любой вариант.

Минимально адекватный учитель (MAT)

MAT отвечает на два типа запросов к оракулу.

- Membership (принадлежность): $\omega \mapsto \omega \in \mathscr{L}$. Возвращает 1, если ω принадлежит $\mathscr{L}$, и 0 иначе.
- Equivalence (эквивалентность): $\mathscr{A} \equiv \mathscr{L}$ (т.е. принимает на вход описание регулярного языка, например, конечным автоматом). Возвращает либо сообщение о том, что эквивалентность выполнена, либо контрпример: слово ω такое, что либо $\omega \in \mathscr{L}(\mathscr{A})$, но $\omega \notin \mathscr{L}$, либо $\omega \in \mathscr{L}$, но $\omega \notin \mathscr{L}(\mathscr{A})$.

Расширенная таблица наблюдений

Алгоритмы $\mathtt{L}^*$ и $\mathtt{NL}^*$ строят описание КА при условии обращения к MAT посредством постепенных приближений таблицы классов эквивалентности. На каждом этапе вычислений таблица должна удовлетворять следующим свойствам:

- полнота — при заглядывании «на шаг вперёд» не должно получаться строк, демонстрирующих иное поведение относительно уже существующих.
- непротиворечивость — если два префикса демонстрируют согласованное поведение в таблице, то при заглядывании «на шаг вперёд» они также должны вести себя согласованно.

Чтобы осуществить требуемое заглядывание, таблица строится из двух частей. Основная состоит из множества строк $\mathcal{S}$ и столбцов $\mathcal{E}$, содержимое таблицы — отметки, принадлежат ли uv языку $\mathscr{L}$, где $u \in \mathcal{S}$, $v \in \mathcal{E}$. Расширенная часть — это строки из $\mathcal{S}$, не являющиеся префиксами других строк из $\mathcal{S}$, с приписанными к ними элементами алфавита Σ, и всё те же столбцы $\mathcal{E}$.

Общая канва алгоритмов $\mathtt{L}^*$ и $\mathtt{NL}^*$

1. Если таблица $\mathcal{S} \times \mathcal{E}$ неполна, то существует элемент из $\mathcal{S}.\Sigma$, который порождает новую строку в таблице. Добавляем его в $\mathcal{S}$ и обновляем расширенную таблицу.
2. Если $\mathcal{S} \times \mathcal{E}$ противоречива, то существует суффикс ν из $\mathcal{E}$, показывающий различное поведение строк u_1, u_2 при приписывании к ним суффикса $\gamma\nu$ ($\gamma \in \Sigma$). Добавляем суффикс $\gamma\nu$ в $\mathcal{E}$ и достраиваем таблицу.
3. Когда таблица полна и непротиворечива, строим по ней описание регулярного языка и делаем запрос к МАТ на эквивалентность. Если МАТ вернул контрпример — добавляем его и все его префиксы в множество $\mathcal{S}$ и продолжаем процесс. Вариант: добавить контрпример и все его суффиксы в $\mathcal{E}$.
4. Результат — описание регулярного языка, которое признаётся оракулом как корректное.

Алгоритм L^*

- Условие полноты — отсутствие в $\mathcal{S}.\Sigma \times \mathcal{E}$ строк, которые отличаются от строк в $\mathcal{S} \times \mathcal{E}$.
- Условие непротиворечивости — отсутствие в $\mathcal{S}.\Sigma \times \mathcal{E}$ таких позиций i, j, k, что $u_i v_k \neq u_j v_k$, притом что $u_i = u_i'\gamma$, $u_j = u_j'\gamma$, и строки в таблице $\mathcal{S} \times \mathcal{E}$ для u_i' и u_j' совпадают.
- ДКА по таблице строится следующим образом.
    - Состояния ДКА — кратчайшие слова из $\mathcal{S}$, порождающие разные строки в $\mathcal{S} \times \mathcal{E}$.
    - Начальное состояние соответствует префиксу ε.
    - Конечные состояния — те, которые содержат 1 в столбце, помеченном ε.
    - Если $u\gamma \equiv u'$ (т.е. $u\gamma$ и u' соответствуют одинаковым строчкам в таблице), то $\langle u, \gamma \rangle \to u'$ добавляется в ДКА как переход.

Алгоритм $\mathtt{NL}^*$

Алгоритм строит минимальный остаточный НКА (RFSA).
Скажем, что строка r накрывается $\bigcup_k r_k$, если $\forall i(r[i] = \bigvee_k r_k[i])$ (т.е. она является поэлементной дизъюнкцией строк r_k).
Строка r_1 поглощает r_2 ($r_2 \sqsubseteq r_1$), если $\forall i(r_1[i] \geqslant r_2[i])$.

- Условие полноты — отсутствие в $\mathcal{S}.\Sigma \times \mathcal{E}$ строк, которые не накрываются строками в $\mathcal{S} \times \mathcal{E}$.
- Условие непротиворечивости — отсутствие в $\mathcal{S} \times \mathcal{E}$ таких позиций i, j, k, что $u_i \sqsubseteq u_j$, но для некоторого $\gamma \in \Sigma$ $u_i\gamma \not\sqsubseteq u_j\gamma$ в расширенной таблице.
- НКА по таблице строится следующим образом.
    - Состояния НКА — кратчайшие слова из $\mathcal{S}$, базисные (т.е. не накрывающиеся набором других) в $\mathcal{S} \times \mathcal{E}$.
    - Начальные состояния включают строки, поглощаемые строкой ε (чтобы перейти к классическому НКА, придётся стянуть их в одно).
    - Конечные состояния — те, которые содержат 1 в столбце, помеченном ε.
    - Если $v \sqsubseteq u\gamma$, то $\langle u, \gamma \rangle \to v$ добавляется в НКА как переход.