Формальные языки и трансляции
3 семестр
Лектор: Ахтямов П.И.
осень 2021

Авторы билетов (лучшие котики):

Спицын Николай

Савичев Дмитрий

Подзорова Полина

Чубенко Полина

Клячин Артемий

Климанова Ирина

Сбродов Егор

Содержание

1. Автоматы и регулярки

1. Недетерминированные конечные автоматы (НКА). Различные варианты определений. Автоматные языки.

Определение: Алфавит Σ — непустое конечное множество, элементы которого называются символами. При этом Σ^* — множество слов, состоящее из всех слов Σ, $\varepsilon \in \Sigma^*$.

Определение: Формальный язык L — некоторое подмножество Σ^*.

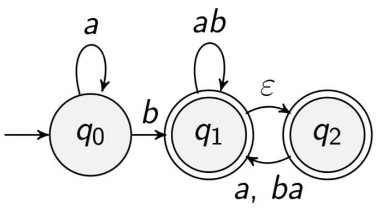
Определение: Недетерминированный конечный автомат (НКА) — кортеж $M = \langle Q, \Sigma, \Delta, q_0, F \rangle$:

1. Q — множество состояний, Q — конечное множество, то есть $|Q| < \infty$;
2. Σ — алфавит;
3. $\Delta \subset Q \times \Sigma^* \times Q$ — множество переходов *(т.е. состояние$_1$ $\xrightarrow{слово}$ состояние$_2$)*;
4. $q_0 \in Q$ — стартовое состояние;
5. $F \subset Q$ — множество завершающих состояний.

Определение: Конфигурация в автомате $\langle Q, \Sigma, \Delta, q_0, F \rangle$ — элемент $\langle q, w \rangle \in Q \times \Sigma^*$.

Определение: Отношение $\vdash$ достижимости по M — наименьшее рефлексивное транзитивное отношение над $Q \times \Sigma^*$, такое что:

1. $\forall w \in \Sigma^* : (\langle q_1, w \rangle \to q_2) \in \Delta \Longrightarrow \langle q_1, w \rangle \vdash \langle q_2, \varepsilon \rangle$
2. $\forall u, v \in \Sigma^* : \langle q_1, u \rangle \vdash \langle q_2, \varepsilon \rangle, \ \langle q_2, v \rangle \vdash \langle q_3, \varepsilon \rangle \Longrightarrow \langle q_1, uv \rangle \vdash \langle q_3, \varepsilon \rangle$
3. $\forall u \in \Sigma^* : \langle q_1, u \rangle \vdash \langle q_2, \varepsilon \rangle \Longrightarrow \forall v \in \Sigma^* \langle q_1, uv \rangle \vdash \langle q_2, v \rangle$



$M = \langle Q, \Sigma, \Delta, q_0, F \rangle$

Пример: Рассмотрим как автомат с картинки распознает слово $abab$:
$\langle q_0, abab \rangle \vdash \langle q_0, bab \rangle \vdash \langle q_1, ab \rangle \vdash \langle q_1, \varepsilon \rangle$
По транзитивности получаем: $\langle q_0, abab \rangle \vdash \langle q_1, \varepsilon \rangle$

Поясним переходы:
Первый и второй $\vdash$ работают по свойству 3, а третий $\vdash$ работает по свойству 1.

Определение: Для автомата $M = \langle Q, \Sigma, \Delta, q_0, F \rangle$ языком $L(M)$, задаваемым автоматом M, является множество $\{w \in \Sigma^* \mid \exists q \in F : \langle q_0, w \rangle \vdash \langle q, \varepsilon \rangle\}$.

Определение: Язык L - автоматный, если существует такой НКА M, что $L = L(M)$.

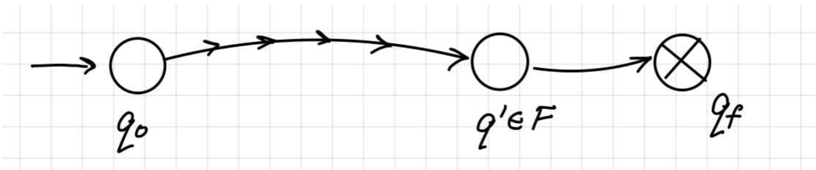
Утверждение об НКА с одним завершающим состоянием: Для любого автоматного языка L существует НКА $M' = \langle Q', \Sigma, \Delta', q_0', F' \rangle$, такой что $L(M') = L$, и $|F'| = 1$.

Идея доказательства: добавим одно завершающее состояние q_f на замену остальным и добавим ε-переходы из "предыдущих" завершающих состояний в новое.

Введем обозначения: $\vdash_i$ - достижимость за i переходов

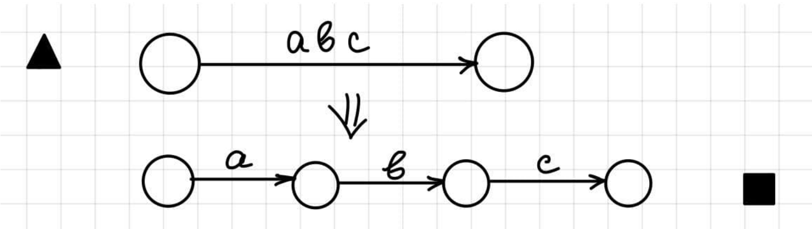
▲ L — автоматный язык, значит, существует НКА $M = \langle Q, \Sigma, \Delta, q_0, F \rangle$, такой что $L(M) = L$. Введём $M' = \langle Q \cup \{q_f\}, \Sigma, \Delta', q_0, \{q_f\} \rangle$, где $\Delta' = \Delta \cup \{\langle q, \varepsilon \rangle \mapsto q_f \mid q \in F\}$. Далее нужно доказать, что $L(M) = L(M')$.

Докажем, что $L(M) \subset L(M')$. По определению из того, что $w \in L(M)$, следует, что существует состояние $q \in F$, что $\langle q_0, w \rangle \vdash \langle q, \varepsilon \rangle$. В автомате M' есть переход $\langle q, \varepsilon \rangle \mapsto q_f \implies \langle q, \varepsilon \rangle \vdash \langle q_f, \varepsilon \rangle$. Так как $\langle q_0, w \rangle \vdash \langle q, \varepsilon \rangle \vdash \langle q_f, \varepsilon \rangle$, то $w \in L(M')$.



Докажем, что $L(M) \supset L(M')$. Из того, что $w \in L(M')$, следует, что $\langle q_0, w \rangle \vdash \langle q_f, \varepsilon \rangle$. Но так как в q_f можно добраться только по ε-переходу, то существует состояние q', что $\langle q_0, w \rangle \vdash \langle q', \varepsilon \rangle \vdash_1 \langle q_f, \varepsilon \rangle \implies q' \in F$. А в автомате M $\langle q_0, w \rangle \vdash \langle q', \varepsilon \rangle$. Значит, $w \in L(M)$. ■

Утверждение об НКА с не более однобуквенными переходами: Для любого автоматного языка L существует НКА $M = \langle Q, \Sigma, \Delta, q_0, F \rangle$, такой что $L = L(M)$ и $\forall (\langle q_1, w \rangle \mapsto q_2) \in \Delta$ $|w| \leqslant 1$



Теорема об НКА с однобуквенными переходами: Для любого НКА $M = \langle Q, \Sigma, \Delta, q_0, F \rangle$ существует НКА $M' = \langle Q, \Sigma, \Delta', q_0, F' \rangle$, такой что $L(M) = L(M')$ и:

$$\forall (\langle q_1, w \rangle \to q_2) \in \Delta \quad |w| = 1$$

▲ Обозначим множество вершин, достижимых из q по w как $\Delta(q, w) = \{q' | \langle q, w \rangle \vdash \langle q', \varepsilon \rangle\}$. Считаем, что в любом переходе $|w| \leqslant 1$. Введём следующие множества:

$$F' := \{q \mid \Delta(q, \varepsilon) \cap F \neq \varnothing\}$$
$$\Delta' = \{\langle q_1, a \rangle \to q_2 \mid \exists q_3 \in \Delta(q_1, \varepsilon) : \langle q_3, a \rangle \to q_2\}$$

Докажем, что $L(M') = L(M)$

Пусть $w \in L(M')$. Тогда $\exists q' \in F' : \langle q_0, w \rangle \vdash_{M'} \langle q', \varepsilon \rangle$.

Тогда $\exists q \in F : \Delta(q', \varepsilon) = q \implies \langle q', \varepsilon \rangle \vdash_M \langle q, \varepsilon \rangle$.

Рассмотрим $w = w_1 w_2 \dots w_n$. Тогда верно следующее:

$$\forall m \ \exists q'_m \ : \ \left\langle q'_{m-1}, w_m \right\rangle \to q'_m \in \Delta' \ \ (q'_m := q') \quad (1)$$

Значит, $\exists q_m \ : \ \Delta(q'_{m-1}, \varepsilon) = q_m, \ \langle q_m, w_m \rangle \to q'_m$ и $\left\langle q'_{m-1}, w_m \right\rangle \vdash_M \langle q'_m, \varepsilon \rangle \quad (2)$

Из (1) и (2) следует, что

$$\langle q'_0, w_1 \dots w_m \rangle \vdash_M \langle q'_1, w_2 \dots w_m \rangle \vdash_M \langle q'_2, w_3 \dots w_m \rangle \dots \vdash_M \langle q'_m, w_m \rangle \vdash_M \langle q, \varepsilon \rangle$$

Следовательно, $w \in L(M)$.

В обратную сторону: $w \in L(M) \Rightarrow \exists q \in F : \ \langle q_0, w \rangle \vdash_M \langle q, \varepsilon \rangle$. Пусть $w = w_1 w_2$ (для больших n аналогично). Тогда есть цепь

- $\langle q_0, w_1 w_2 \rangle \vdash_M \langle q'_1, w_1 w_2 \rangle$
- $\langle q'_1, w_1 w_2 \rangle \vdash_{M,1} \langle q_1, w_2 \rangle$ (читаем символ w_1)
- $\langle q_1, w_2 \rangle \vdash_M \langle q'_2, w_2 \rangle$
- $\langle q'_2, w_2 \rangle \vdash_{M,1} \langle q_2, \varepsilon \rangle$ (читаем символ w_2)
- $\langle q_2, \varepsilon \rangle \vdash_M \langle q, \varepsilon \rangle$

Получаем, что $\Delta(q_0, \varepsilon) = q'_1, \ \langle q'_1, w_1 \rangle \to q_1 \Rightarrow \langle q_0, w_1 \rangle \to q_1 \in \Delta'$.

Аналогично $\langle q_1, w_2 \rangle \to q_2 \in \Delta'$. Так как $\Delta(q_2, \varepsilon) = q \in F \Rightarrow q_2 \in F'$.

Тогда $\langle q_0, w_1 w_2 \rangle \vdash_{M'} \langle q_1, w_2 \rangle \vdash_{M'} \langle q_2, \varepsilon \rangle$ и $w = w_1 w_2 \in L(M')$ $\blacksquare$

2. Детерминированные конечные автоматы (ДКА). Эквивалентность ДКА и НКА.

Опеределение: НКА $M = \langle Q, \Sigma, \Delta, q_0, F \rangle$ называется детермированным (ДКА), если выполнено:

1. $\forall (\langle q_1, w \rangle \to q_2) \in \Delta' : |w| = 1$
(все переходы являются однобуквенными);

2. $\forall a \in \Sigma,\ q \in Q\ |\Delta(q, a)| \leqslant 1$
(из одного состояния по одному символу можно перейти не более, чем в одно состояние);

Теорема о эквивалентности ДКА и НКА Для любого НКА $M = \langle Q, \Sigma, \Delta, q_0, F \rangle$ существует ДКА M', такой что $L(M) = L(M')$.

▲ Обозначим $\Delta(S, w) = \bigcup\limits_{q \in S} \Delta(q, w)$, где $w \in \Sigma^*$, $S \subset Q$. Построим ДКА $M' = \langle 2^Q, \Sigma, \Delta', \{q_0\}, F' \rangle$, где:

1. $F' = \{S \subset Q \mid S \cap F \neq \varnothing\}$;

2. $\Delta' = \{\langle S, a \rangle \to \Delta(S, a)\}$.

Чтобы понять, что из себя представляет новое множество переходов Δ', докажем следующую лемму:

Лемма: $\Delta'(\{q_0\}, w) = \Delta(\{q_0\}, w)$, где слева – вершина в M', а справа – подмножество в M.

Докажем лемму индукцией по длине слова w.

База. $w = \varepsilon$, тогда $\Delta(\{q_0\}, \varepsilon) = \{q_0\} = \Delta'(\{q_0\}, \varepsilon)$, так как все переходы в автомате M' являются однобуквенными.

Переход. $w = ua$, где $a \in \Sigma$, $u \in \Sigma^*$.

Сначала покажем, что $\Delta(\{q_0\}, ua) = \Delta(\Delta(\{q_0\}, u), a)$:

$$\Delta(\{q_0\}, ua) = \{q \mid \langle q_0, ua \rangle \vdash \langle q, \varepsilon \rangle\} \text{ из опеределения } \Delta$$

По однобуквенности переходов и транзитивности:

$$\{q \mid \langle q_0, ua \rangle \vdash \langle q, \varepsilon \rangle\} = \{q \mid \exists q' : \langle q_0, ua \rangle \vdash \langle q', a \rangle \vdash \langle q, \varepsilon \rangle\} = \{q \mid \exists q' \in \Delta(\{q_0\}, u) : \langle q', a \rangle \vdash \langle q, \varepsilon \rangle\} = \Delta(\Delta(\{q_0\}, u), a)$$

По предположению индукции:

$$\Delta(\Delta(\{q_0\}, u), a) = \Delta(\Delta'(\{q_0\}, u), a)$$

$$S := \Delta'(\{q_0\}, u)$$

$$\Delta(S, a) = \Delta'(S, a) \text{ (следует из определения переходов в ДКА)}$$

$$\Delta'(\Delta'(\{q_0\}, u), a) = \Delta'(\{q_0\}, ua)$$

Лемма доказана.

Теперь покажем, что $w \in L(M) \Longleftrightarrow w \in L(M')$.

$$w \in L(M) \Longleftrightarrow \exists q \in F : \langle q_0, w \rangle \vdash \langle q, \varepsilon \rangle \Longleftrightarrow \Delta(q_0, w) \cap F \neq \varnothing \Longleftrightarrow \Delta(\{q_0\}, w) \cap F \neq \varnothing \overset{lemma}{\Longleftrightarrow} \Delta'(\{q_0\}, w) \cap F \neq \varnothing$$

$$T := \Delta'(\{q_0\}, w)$$

$$T \cap F \neq \varnothing \Longleftrightarrow T \in F',\ \Delta'(q_0', w) \in F',\ q_0' = \{q_0\}$$

$$T \in F' \Longleftrightarrow w \in L(M') \qquad \blacksquare$$

3. Свойства класса автоматных языков. Замкнутость относительно булевых операций.

Опеределение: Полный ДКА. Полный ДКА (ПДКА) - ДКА, для которого выполнено:

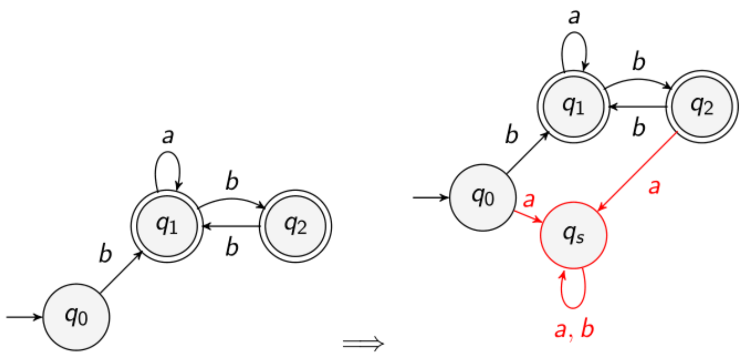
$$\forall a \in \Sigma, q \in Q \ \ |\Delta(q, a)| = 1$$

Утверждение: Для любого автоматного языка L существует ПДКА M, такой что $L(M) = M$ (т.е. автоматы распознают одинаковое множество слов);

Метод построения ПДКА из ДКА:

1) строим "стоковую" вершину.

2) Добавляем из всех вершин переходы по недостающим буквам в "сток".



Появятся ли новые слова? - нет, потому что, если мы попали в стоковую вершину, то не сможем "выбраться" из неё.

Опеределение: Итерация Клини для языка L.

$$L^* = \cup_{k=0}^{\infty} L^k$$

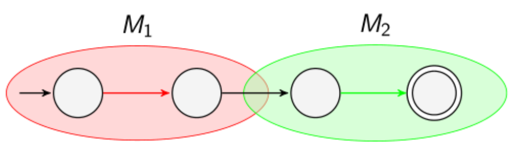
Теорема: Класс автоматных языков замкнут относительно

1. Конкатенации
2. Объединения
3. Пересечения
4. Итерации Клини
5. Дополнения

▲ Далее будем рассматривать только НКА с одним завершающим состоянием. Для того чтобы после операции у итогового автомата было одно завершающее состояние, добавляем состояние и соединяем завершающие состояния с ним с помощью ε-переходов. (делаем новое состояние - завершающим, а старые - не завершающими)

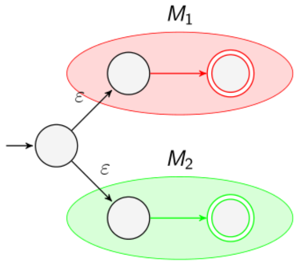
1) Конкатенация M_1 и M_2:

Соединяем ε-переходами завершающее состояние M_1 со стартовыми состояниями M_2.



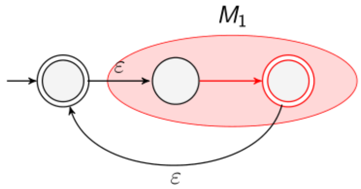
2) Объединение M_1 и M_2:

Добавляем стартовое состояние. Соединяем её со стартовыми состояниями M_1 и M_2 с помощью ε-переходов.



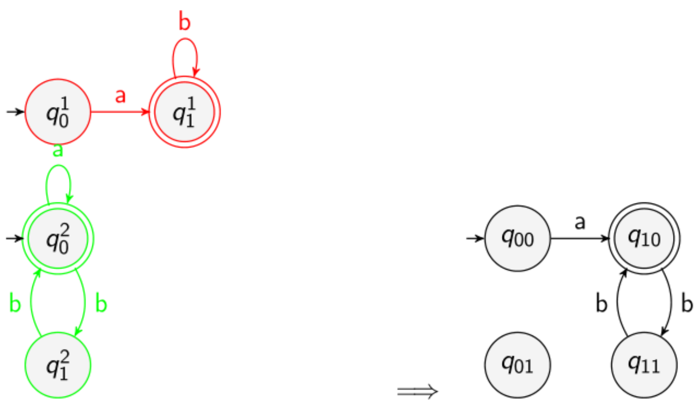
4) Итерации Клини над M_1:

Добавляем стартово-завершающее состояние. С помощью ε-переходов соединяем её с начальными состояниями M_1, а завершающее состояния M_1 с ней.



3) Пересечение M_1, M_2:

Строим "декартово произведение"автоматов с одно буквенными переходами.



То есть пересечение будет состоять из состояний, каждому из которых соответствует пара чисел (i, j), это номера состояний из M_1 и M_2 соответственно, которым это состояние соответствует. И между состояниями (i_1, j_1) и (i_2, j_2) будет проходить ребро с символом k, если между i_1 и i_2 проходило ребро с символом k в M_1 и между j_1 и j_2 проходило ребро с символом k в M_2. (i, j) - стартовое состояние, если i - стартовое в M_1, j - стартовое в M_2. Аналогично с завершающем состоянием.

5) Дополнение: строим ПДКА и инвертируем терминальность всех состояний.

4. Регулярные выражения. Теорема Клини о совпадении классов регулярных и автоматных языков. Регулярный автомат, алгоритм построения.

Введем обозначения:
Regex (регулярное выражение) обозначим за R,
Language (язык) – за L,
$L(R_i)$ (язык, который задается регулярным выражением R) – L_i.

Опеределение: Рекурсивное определение регулярного выражения.

$Regex(R)$	$Language(L_i = L(R_i))$
0	$\varnothing$
1	$\{\varepsilon\}$
$a,\ a \in \Sigma$	$\{a\}$
$R_1 + R_2$	$L_1 \cup L_2$
$R_1 \cdot R_2$	$L_1 \cdot L_2$
R^*	L^*

Здесь ε – пустое слово, «$\cdot$» – операция конкатенации языков (в полученном языке $L_1 \cdot L_2$ лежат слова вида a_1a_2, где слово a_1 лежит в языке L_1, а слово a_2 лежит в языке L_2), «$*$» – звезда Клини.

Напомним определение звезды Клини: $V^* = \bigcup_{i=0}^{\infty} V^i$

Приоритет операций в регулярных выражениях (левее — приоритетнее): $* \to \cdot \to +$

Опеределение: Язык L – регулярный, если он задается регулярным выражением.

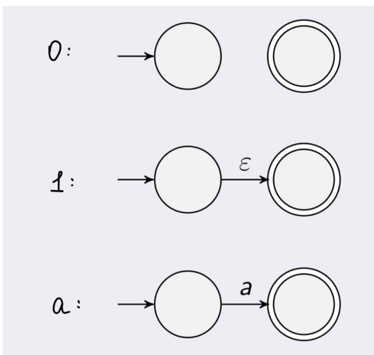
Теорема Клини: Классы регулярных и автоматных языков совпадают.

▲ Докажем два вложения:

1. Регулярные $\subseteq$ Автоматные

Индукция по построению выражения. Немного изменим утверждение – докажем, что по регулярному выражению можно построить НКА с 1 завершающим состоянием, который задает тот же язык.

База: Построим автоматы для регулярных выражений: $0, 1, a \in \Sigma$.



Переход:

1) $R = R_1 + R_2$. Построим автомат A_1 для R_1, для которого вершина S_1 – стартовая, а вершина F_1 – единственная терминальная. Для R_2 это будут автомат A_2 со стартовой вершиной S_2 и терминальной F_2. Создадим новую вершину S, которая и будет стартовой в новом автомате. Из нее проведем два ребра с ε-переходами в S_1 и в S_2. Аналогично соединим завершающие в автоматах с новой завершающей вершиной F. Нетрудно доказать, что такой автомат задаст тот же язык, что и наше регулярное выражение.

2) $R = R_1 \cdot R_2$. Аналогично прошлому пункту получим автоматы для R_1 и R_2 с теми же обозначениями. Вершина S_1 будет стартовой в нашем новом автомате. Добавим также ε-переход из F_1 в S_2, уберем терминальность F_1.

3) $R = R_1^*$. Построим автомат A_1 для R_1 со стартовой вершиной S_1 и терминальной вершиной F_1. Создадим вершину S – новую стартовую вершину, пометим ее терминальной. Добавим из нее и из F_1 ε-переход в S_1.

2. Автоматные $\subseteq$ Регулярные

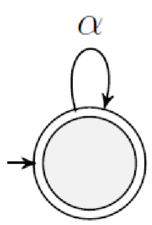
Замечание: Регулярный автомат – НКА, в котором на ребрах записаны регулярные выражения. Докажем утверждение для регулярных автоматов.

Замечание: Всякий НКА задается регулярным автоматом с 1 завершающим состоянием.

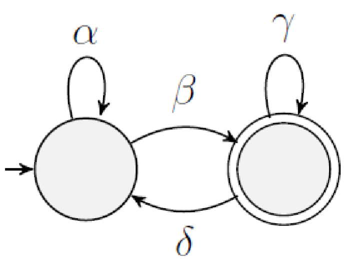
Индукция по $|Q|$ (количеству состояний – вершин) в регулярном автомате.

База:

1) $|Q| = 1$. Тогда в регулярном автомате стартовое состояние является завершающим, и можно однозначно построить регулярное выражение. Такому автомату соответсвует регулярное выражение a^*

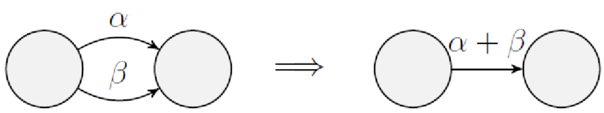


2) $|Q| = 2$. Стартовое состояние и завершающее состояние различны, и можно тоже однозначно построить регулярное выражение. Такому автомату соответсвует регулярное выражение $\alpha^*\beta(\gamma + \delta\alpha^*\beta)^*$

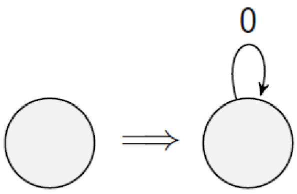


Для случая, когда завершающее состояние – это начальная вершина, регулярное выражение будет $(\gamma + \delta\alpha^*\beta)^*$.

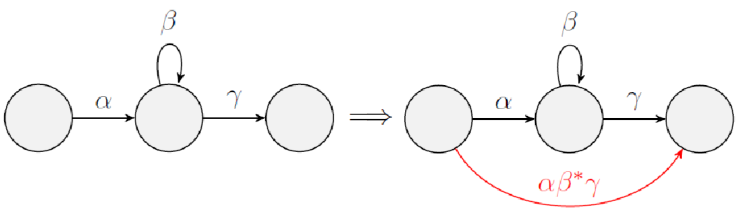
Переход: **Замечание:** Есть нестартовая и незавершающая вершина!

1) Удаляем кратные ребра: 

Кратные ребра означают, что мы можем выбрать, какой символ будем использовать. Именно этот смысл и несет в себе операция «+».

2) Добавляем циклы на себя: 

3) Удаляем нестартовое и незавершающее состояние:



Теперь у нас на одно сотояние стало меньше, т.е. мы можем воспользоваться утверждением индукции. ■

5. Минимальный ДКА, его существование.

Мотивировка: может быть много состояний. А еще не очень понятно, как сравнивать два автомата на эквивалентность. Вернее, если проверять «в лоб», будет долго. Один из способов решить эти проблемы – минимизация автомата.

Пусть $L \subset \Sigma^*$ – автоматный язык, M – ПДКА для L.

Опеределение: Минимальный ПДКА M, распознающий язык L, если M – минимальный по количеству состояний.

Опеределение: Определим отношение эквивалентности $\sim_L$ на Σ^* :

$$u \sim_L v \Longleftrightarrow \forall w \in \Sigma^* \ (uw \in L \Longleftrightarrow vw \in L)$$

Определение корректно (рефлексивность, симметричность и транзитивность очевидны).

Множество классов эквивалентности в этом случае: $\Sigma^*/_{\sim_L} := \{\{u | u \sim_L v\} \mid v \in \Sigma^*\}$

Опеределение: Определим отношение эквивалентности $\sim_M$ на Q :

$$q_1 \sim_M q_2 \Longleftrightarrow \forall w \in \Sigma^* \ (\Delta(q_1, w) \in F \Longleftrightarrow \Delta(q_2, w) \in F)$$

Если $q_1 \sim_M q_2$, то состояния можно объединить.

Напоминание: Множество вершин, достижимых из q по w – $\Delta(q, w) = \{q' | \langle q, w\rangle \vdash \langle q', \varepsilon\rangle\}$..

Лемма Пусть $L_q := \{w \,|\, \Delta(q_0, w) = q\}$. Тогда каждый класс эквивалентности в нашем фактор-множестве являетеся объединением классов в L_q.

▲ Возьмем слово $u \in [u] \in \Sigma^*/_{\sim_L}$, где $[u]$ – это класс эквивалентности для u. Рассмотрим путь по u из q_0, а именно, $q_u = \Delta(q_0, u)$. Для любого слова $w \in [u]$, $q_w = \Delta\,(q_0, w)$. Тогда $[u] = \bigcup\limits_{q_w, w \in [u]} L_{q_w}$. Далее докажем, почему это так.

Пусть $v \in [u] \implies v \sim_L u \implies q_v = \Delta\,(q_0, v) \implies v \in L_{q_v}$. Тогда $v \in \bigcup\limits_{q_w, w \in [u]} L_{q_w}$.

Пусть $v \in \bigcup\limits_{q_w, w \in [u]} L_{q_w}$. Тогда существует состояние q_z, $z \in [u]$, что $v \in L_{q_z} = \{w | \Delta\,(q_0, w) = q_z\}$.

$$z \in [u] \Longrightarrow z \sim_L u \overset{def}{\Longrightarrow} \forall w \in \Sigma^* \ (zw \in L \Longleftrightarrow uw \in L)$$

$$v \in L_{q_z} \Longrightarrow \left.\begin{aligned} \Delta\,(q_0, v) &= q_z \\ \Delta\,(q_0, z) &= q_z \end{aligned}\right\} \quad \Longrightarrow \quad v \sim_L z \text{ так как } \forall w \in \Sigma^* \ \Delta\,(q_0, vw) \overset{(*)}{=} \Delta\,(q_0, zw)$$

$(*)$: $\Delta\,(q_0, vw) = \Delta\,(\Delta(q_0, v), w) = \Delta\,(q_z, w) = \Delta\,(\Delta(q_0, z), w) = \Delta\,(q_0, zw)$

Так как $v \sim_L z$, $z \in [u]$, то $v \in [u]$. Значит, $[u] = \bigcup\limits_{q_w, w \in [u]} L_{q_w}$, и каждый класс эквивалентности из $\Sigma^*/_{\sim_L}$ — объединение классов в L_q. ■

Следствие $|\Sigma^*/_{\sim_L}| \leq |Q|$

Теперь перейдем к минимальному ПДКА, а именно докажем его существование (тут) и единственность с точностью до изоморфизма (билет 6).

Лемма Для любого автоматного языка L существует ПДКА M' такой, что все состояния в M' попарно неэквивалентны.

Что необходимо доказать?

1) Переходы и завершающие состояния согласованы

2) Распознаваемые языки совпадают

3) Состояния попарно неэквивалентны

▲ Рассмотрим автомат над классами эквивалентности $\sim_M$. Класс эквивалентности q обозначим за $[q]$. $M' = \langle Q/_{\sim_M}, \Sigma, \Delta', [q_0], F' \rangle$, где:

$$\Delta' = \{\langle [q_1], a \rangle \to [q_2] \mid \exists \langle q_1, a \rangle \to q_2 \in \Delta\}$$
$$F' = \{[q_f] \mid q_f \in F\}$$

1) Проверим, что множества Δ', F' заданы корректно.

Для Δ': Пусть $q_1 \sim_m q_1'$, и существует a такое, что $\Delta(q_1, a) \nsim_m \Delta(q_1', a)$.

$$q_1 \sim_M q_1' \overset{def}{\Longrightarrow} (\forall w \in \Sigma^* \ \ \Delta(q, w) \in F \Longleftrightarrow \Delta(q_1', w) \in F)$$
$$w = au \Longrightarrow (\forall u \in \Sigma^* \ \ \Delta(q_1, au) \in F \Longleftrightarrow \Delta(q_1', au) \in F)$$

Далее обозначим $\Delta(q_1, a) = q_2$, а $\Delta(q_1', a) = q_2'$.

$$\Delta(q_1, au) = \Delta(\Delta(q_1, a), u) = \Delta(q_2, u)$$
$$\Delta(q_1', au) = \Delta(q_2', u)$$
$$(\Delta(q_2, u) \in F \Longleftrightarrow \Delta(q_2', u) \in F) \ \Longrightarrow q_2 \sim_m q_2'.$$

Приходим к противоречию.

Для F':

$$q_1 \in F,\, q_2 \sim_M q_1 \overset{w=\varepsilon}{\Longrightarrow} (\Delta(q_1, \varepsilon) \in F \Longleftrightarrow \Delta(q_2, \varepsilon) \in F)$$
$$\text{Значит, } q_1 \in F \Longleftrightarrow q_2 \in F$$

2) Теперь покажем, что $L(M) = L(M')$.

Для этого нужно показать, что $w \in L(M) \Longleftrightarrow \Delta(q_0, w) \in F \overset{?}{\Longleftrightarrow} \Delta([q_0], w) \in F'$.

Докажем утверждение: $\forall u : \Delta(q_0, u) = q_1 \Longleftrightarrow \Delta([q_0], u) = [q_1]$.

Индукция по длине слова u.

База. $|u| = 0 \Longrightarrow u = \varepsilon$. Тогда $\Delta(q_0, \varepsilon) = q_0$, $\Delta([q_0], \varepsilon) = [q_0]$.

Переход. Пусть $u = va$, $v \in \Sigma^*$, $a \in \Sigma$.

$$\Delta(q_0, va) = q_1 \Longrightarrow \exists q_2 \ \Delta(q_0, v) = q_2, \ \Delta(q_2, a) = q_1$$

По предположению индукции, $\Delta([q_0], u) = [q_2]$, $\Delta([q_2], a) = [q_1]$, так как переход $\langle q_2, a\rangle \to q_1 \in \Delta$ тогда и только тогда, когда $\langle [q_2], a\rangle \to [q_1] \in \Delta'$. По транзитивности, $\Delta([q_0], ua) = [q_1]$.

3) Теперь покажем, что состояния попарно неэквивалентны. В автомате, построенном на классах эквивалентности состояний никакие два состояния не эквивалентны, потому что тогда бы они лежали в одном классе, т.е. были бы одним состоянием.

Пусть $[q_1] \sim_{M'} [q_2]$. Тогда $\forall w : \Delta_{M'}([q_1], w) \in F' \Longleftrightarrow \Delta_{M'}([q_2], w) \in F'$ по определению.

$$[q_{1f}] = \Delta_{M'}([q_1], w) \in F'$$
$$[q_{2f}] = \Delta_{M'}([q_2], w) \in F'$$
$$\exists q_{1f} \in F : \Delta_M(q_1, w) = q_{1f} \in F \Longleftrightarrow \exists q_{2f} \in F : \Delta_M(q_2, w) = q_{2f} \in F$$
$$q_1 \sim_M q_2 \Longrightarrow [q_1] = [q_2]$$

■

Теорема: M — минимальный ПДКА, распознающий язык L, тогда и только тогда, когда любые два состояния попарно неэквивалентны и все состояния достижимы из стартового.

Теперь запишем более формально:

$$M \text{ — минимальный ПДКА} \Longleftrightarrow \begin{cases} \forall q_1, q_2 \in Q \; q_1 \nsim q_2 \\ \forall q \in Q \; \exists w \in \Sigma^* : \; \langle q_0, w\rangle \vdash \langle q, \varepsilon\rangle \end{cases}$$

▲

$\Longrightarrow$ Если $q_1 \sim_M q_2$, то $[q_1] = [q_2]$, и их можно объединить в одно состояние, значит, M не был бы минимальным, и тогда из минимальности следует, что $q_1 \nsim_M q_2$. Если среди состояний есть недостижимые, то если их удалить, то множество принимаемых слов не изменится.

$\Longleftarrow$ По следствию из леммы о L_q: $|\Sigma^*/_{\sim_L}| \leqslant |Q|$. Рассмотрим w_1, w_2 такие, что $\Delta(q_0, w_1) \neq \Delta(q_0, w_2)$. Введём обозначения:

$$\Delta(q_0, w_1) = q_1$$
$$\Delta(q_0, w_2) = q_2$$

Неэквивалентность состояний q_1, q_2 означает, что существует слово w, что б.о.о:

$$\Delta(q_1, w) = \Delta(q_0, w_1 w) \in F \Longleftrightarrow w_1 w \in L$$
$$\Delta(q_2, w) = \Delta(q_0, w_2 w) \notin F \Longleftrightarrow w_2 w \notin L$$
$$\text{Следовательно, получили что } w_1 \nsim_L w_2$$

Тогда для автомата M со множеством состояний Q' выполняется, что $|\Sigma^*/_{\sim_L}| \geqslant |Q'|$, но тогда $|Q| \geqslant |\Sigma^*/_{\sim_L}| \geqslant |Q'|$, и M — минимальный. ■

6. Минимальный ДКА, его единственность.

Определение: M – минимальный ПДКА, распознающий язык L, если M минимальный по количеству состояний.

В предыдущем билете мы доказали существование минимального ПДКА (это логически следует из леммы и теоремы).

Замечание 1: в МПДКА $M = \langle Q, \ldots \rangle \hookrightarrow |\Sigma^*/_{\sim_L}| = |Q|$

Замечание 2: $[u] \in \Sigma^*/_{\sim_L} \Longrightarrow [u] = \cup_q L_q$, где $L_q := \{w \mid \Delta(q_0, w) = q\}$ в МПДКА. Тогда получаем, что $[u] \in \Sigma^*/_{\sim_L} \Longrightarrow \exists!\, q : \; [u] = L_q$.

Теорема: Для любого автоматного языка L существует единственный с точностью до изоморфизма минимальный ПДКА M, такой что $L = L\,(M)$.

▲ Пусть M — минимальный ПДКА, Q_M — множество его состояний.

Построим автомат $M_0 = \langle \Sigma^*/_{\sim_L}, \Sigma, \Delta, [\varepsilon], \{[w] | w \in L\}\rangle$. Для любых $u \in \Sigma^*$, $a \in \Sigma$ верно, что $\Delta\,([u], a) = [ua]$ (факт 1). Так как количество состоянии автомата, соответствующему языку L, конечно, то по следствию из леммы о классах эквивалентности и L_q: $|\Sigma^*/_{\sim_L}| < \infty$.

Факт 1 верен вследствие следующего:

$$u \sim_L v \Longrightarrow ua \sim_L va \Longleftrightarrow \forall w\,(uaw \in L \Longleftrightarrow vaw \in L)$$
$$w' = aw, \forall w'\,(uw' \in L \Longleftrightarrow vw' \in L) \Longleftrightarrow u \sim_L v$$

Так как $u \sim_L v$, то если $u \in L$, то $v \in L$, и наоборот (просто берем $w = \varepsilon$).

Теперь рассмотрим $\psi : Q_M \to \Sigma^*/_{\sim_L}$, $\psi\,(q) = \{w | \Delta\,(q_0, w) = q\} = L_q$. Из замечаний $\Longrightarrow$ ψ – взаимнооднозначное отображение. Более того, покажем, что ψ — изоморфизм между автоматами, как графами. Для этого нужно показать, что:

1. $\Delta\,(\psi(q), a) = [\psi\,(q)\,a]$;
2. $q \in F \Longleftrightarrow \psi\,(q) \subseteq L$.

Покажем, почему выполняется (1).

$$\psi\,(q) = [u],\, \psi\,(q') = [u'],\, \Delta\,(q, a) = q'$$
$$\Delta\,(\psi(q), a) = \Delta\,(\{w | \Delta(q_0, w) = q\}, a) = \{w' = wa | \langle q_0, w' \rangle = q'\} = [wa]$$
$$[u] = [w],\, [u'] = [wa] \text{ по транзитивности переходов в автомате}$$

Покажем, почему выполняется (2). Из того, что $q \in F$, следует, что слова из множества $\psi\,(q) = \{w | \Delta\,(q_0, w) = q\}$ принадлежат языку L, так они распознаются автоматом, поскольку q является завершающим состоянием. А так как $\psi\,(q) = \{w | \Delta\,(q_0, w) = q\} \subseteq L$, то так как они распознаются автоматом, соответствующему языку L, то $q \in F$.

Пусть ψ_1 — изоморфизм между минимальными ПДКА M_1 и M_0, ψ_2 — изоморфизм между минимальными ПДКА M_2 и M_0. Тогда M_1 и M_2 изоморфны между собой — этому соответствует изоморфизм $\psi_2^{-1} \circ \psi_1$, композиция изоморфизмов является изоморфизмом. ■

7. Минимальный ДКА, алгоритм построения.

Мы хотим преобразовать автомат так, чтобы его состояния были попарно неэквивалентны. Но у нас есть только слова. Как быть? Введем эквивалентность по словам фиксированной длины.

Опеределение: $q_1 \underset{n}{\sim} q_2$, если для любого слова $w : |w| \le n$:

$$\Delta(q_1, w) \in F \iff \Delta(q_2, w) \in F$$

. Это отношение эквивалентности (аналогично своему M-аналогу), поэтому можно ввести $Q/_{\underset{n}{\sim}}$.

Лемма: $q_1 \underset{|Q|-2}{\sim} q_2 \Longrightarrow q_1 \sim q_2$

▲ Если $q_1 \underset{i+1}{\sim} q_2 \Longrightarrow q_1 \underset{i}{\sim} q_2$, тогда $|Q/_{\underset{i}{\sim}}| \leqslant |Q/_{\underset{i+1}{\sim}}|$.

Покажем, что если $|Q/_{\underset{i}{\sim}}| = |Q/_{\underset{i+1}{\sim}}|$, то $|Q/_{\underset{i+1}{\sim}}| = |Q/_{\underset{i+2}{\sim}}|$. Пусть существуют состояния q_1 и q_2. такие что $q_1 \underset{i+2}{\nsim} q_2$, $q_1 \underset{i+1}{\sim} q_2$.

Тогда $\exists u, |u| \leqslant i+2$, что без ограничения общности $\Delta(q_1, u) \in F$, $\Delta(q_2, u) \notin F$. Заметим, что должно быть верно, что $|u| = i+2$, иначе возникнет противоречие с $(i+1)$-эквивалентностью. Тогда пусть $u = av$, где a – некоторый символ.

Рассмотрим $p_1 = \Delta(q_1, a), p_2 = \Delta(q_2, a) \Longrightarrow \Delta(p_1, v) \in F,\ \Delta(p_2, v) \notin F$, при этом $|v| \leqslant i+1$

Тогда $p_1 \underset{i+1}{\nsim} p_2$ так как существует слово длины $i+1$, которое их различает $\Longrightarrow p_1 \underset{i}{\nsim} p_2 \Longrightarrow q_1 \underset{i+1}{\nsim} q_2$ Противоречие.

Отсюда следует, что если $|Q/_{\underset{i}{\sim}}| = |Q/_{\underset{i+1}{\sim}}|$, то $|Q/_{\underset{i+1}{\sim}}| = |Q/_{\underset{i+2}{\sim}}|$. И так для всех i. Поэтому если мы остановились, новые классы эквивалентности не появятся. Понятно, что мы можем увеличить класс эквивалентности $\mathrm{O}(|Q|)$ раз

■

Поэтому наш алгоритм выглядит так:

Множество $Q/_{\sim_0}$ представляет собой $\{F, Q \setminus F\}$, то есть это множество из множества завершающих состояний и множества состояний, не являющихся завершающими.

До тех пор, пока количество классов эквивалентности меняется, увеличиваем длину слов, по которым проверяем эквивалентность.

Пусть у нас была длина n, мы перешли к $n+1$. Рассмотрим какой-то класс эквивалентности.

Посмотрим на переход по первой букве. Заметим тогда, что нам останется пройти n символов. Все эти вершины (на расстоянии 1) мы уже распределили по классам эквивалентности на основе слов длины n, поэтому нужно для каждого сотояния найти множество классов эквивалентности его соседей. Если множества различаются, то эти состояния теперь разойдутся по разным классам. Совпадают, значит, останутся в одном классе.

В соответствии с этим получим новое распределение состояний по классам.

8. Лемма о разрастании для автоматных языков. Примеры неавтоматных языков.

Лемма: (о накачке (разрастании))

Пусть L - автоматный язык, $|L| = \infty$. Тогда:

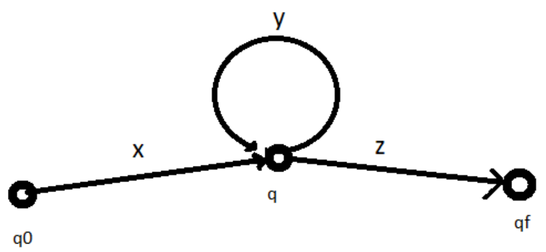
$$\exists P\ \forall w \in L : |w| > P\ (\exists x, y, z : w = xyz, |xy| \leq P, |y| \neq 0 : \forall k \in \mathbb{N} : xy^k z \in L)$$

Идея доказательства:

1) Рассмотрим НКА с однобуквенными переходами M.

2) Взять $P = |Q|$ (количество состояний), найти первый "цикл".

Рассмотрим $w \in L : |w| > P$ => То по принципу Дирихле, найдётся состояние q, которые мы посетили дважды. Образовался цикл ($w = xyz$):



1. $|xy| \leqslant P = |Q|$: допустим, что $|xy| > P = |Q|$, тогда в xy нашёлся бы цикл, где q не будет являться первым пересечением;

2. $|y| \neq 0$, так как среди переходов в автомате M встречаются только однобуквенные, то существует хотя бы одно состояние $q_{new} \neq q$, которое будет посещено.

Слово $xy^k z$ принадлежит языку L для любого $k \in \mathbb{N}$, так как цикл, соответствующий y, может быть повторён k раз. ■

Пример: Язык $\{a^n b^n | n \in \mathbb{N}\}$ не является автоматным.

▲ Воспользуемся отрицанием леммы о разрастании:

$$\forall p\ \exists w \in L : |w| \geqslant P\ \forall x,\ y,\ z : w = xyz, |xy| \leqslant P,\ |y| \neq 0 : \forall k \in \mathbb{N} : xy^k z \notin L$$

. Возьмем произвольное p и рассмотрим $w = a^p b^p$. Тогда

$$w = xyz\ :\ |xy| \leqslant p,\ |y| \geqslant 0\ \Rightarrow\ x = a^k,\ y = a^l,\ l > 0\ \rightarrow\ z = a^{p-k-l} b^p$$

Рассмотрим $xy^2 z = a^k a^{2l} a^{p-k-l} b^p = a^{p+l} b^p$. Так как $p + l \neq p$, значит $xy^2 z \notin L$ ■

9. Алгоритм проверки равенства регулярных выражений. Теорема Майхилла-Нероуда.

Лемма (из билета 6): Для любых $u \in \Sigma^*$, $a \in \Sigma$ верно, что $\Delta\left([u], a\right) = [ua]$

$\blacktriangle$ $u \sim_L v \Longrightarrow \forall w' \left(uw' \in L \Longleftrightarrow vw' \in L\right) \Longrightarrow \forall w \left(uaw \in L \Longleftrightarrow vaw \in L\right) \Longleftrightarrow ua \sim_L va$ $\blacksquare$

Лемма: Для любых $u, v \in \Sigma^*$ верно, что $\Delta\left([u], v\right) = [uv]$

$\blacktriangle$ Индукция по $|v|$. База:

$v = \varepsilon \Longrightarrow \Delta([u], \varepsilon) = [u] = [u\varepsilon]$

$v = a \Longrightarrow \Delta([u], a) = [ua]$ — по предыдущей лемме.

Шаг: $v = v'a$

$$\Delta([u], v'a) = \Delta(\Delta([u], v'), a) = \Delta([uv'], a) = [uv'a] = [uv]$$

$\blacksquare$

Теорема Майхилла-Нероуда: Язык L является автоматным тогда и только тогда, когда $\Sigma^*/_{\sim_L}$ содержит конечное количество классов эквивалентности.

$\blacktriangle$

$\Longrightarrow$ Так как L является автоматным. то для него существует минимальный ПДКА, $|\Sigma^*/_{\sim_L}| = |Q|$, Q - конечно

$\Longleftarrow$ Множество $\Sigma^*/_{\sim_L}$ конечно. Построим канонический ПДКА $M_0 = \langle Q, \Sigma, \Delta, [\varepsilon], F\rangle$, где $Q = \Sigma^*/_{\sim_L}$, $F = \{[w] \mid w \in L\}$, $\Delta = \{\langle [w], a\rangle \to [wa] \mid w \in \Sigma^*, a \in \Sigma\}$. Докажем, что $L(M_0) = L$.

$$w \in L(M_0) \Longleftrightarrow \Delta([\varepsilon], w) \in F \Longleftrightarrow [w] \in F \Longleftrightarrow \exists u \in L\, (w \sim_L u) \Longleftrightarrow w \in L.$$

Заметим, что если $u \in L$, $w \notin L$ то $u \nsim w$. Автомат построен. $\blacksquare$

Алгоритм проверки регулярных выражений на равенство

Пусть R_1 и R_2 — регулярные выражения. Построим по ним минимальные ПДКА M_1 и M_2 соответственно. По теореме о существовании и единственности минимального ПДКА для автоматного языка $L\left(M\right)$ минимальный ПДКА единственен с точности до изоморфизма. Если M_1 и M_2 изоморфны, то регулярные выражения равны, иначе нет.

2. КС-грамматики и МП-автоматы

10. Иерархия Хомского. Праволинейные грамматики. Праволинейные языки. Теорема о совпадении классов автоматных и праволинейных языков.

Иерархия Хомского

$A, B \in N,\ \alpha, \varphi, \psi \in (N \cup \Sigma)^*$, $w \in \Sigma^*$

Грамматики	Правила	Автоматы
Праволинейные	$A \to wB,\ A \to w$	НКА
Контекстно-свободные	$A \to \alpha$	Автоматы с магазинной памятью
Контекстно-зависимые	$\varphi A \psi \to \varphi \alpha \psi$	Линейно-ограниченные недетерминированные автоматы, машины Тьюринга
Порождающие	Любые	Машины Тьюринга

Порождающие грамматики

Опеределение: *Порождающая грамматика*: $G = \{N, \Sigma, P, S\}$, где:

1) N – множество вспомогательных (нетерминальных) символов, $|N| < \infty$.

2) Σ – алфавит – множество (терминальных) символов, $|\Sigma| < \infty$, $\Sigma \cap N = \varnothing$.

3) $S \in N$ – стартовый нетерминал

4) $P \subset (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$ – множество правил, $|P| < \infty$.

Правила грамматики грамматики задаются следующим образом:

$$\alpha \to \beta \in P, \alpha \in (N \cup \Sigma)^+, \beta \in (N \cup \Sigma)^*$$

То есть α должен содержать хотя бы один символ — либо терминальный, либо нетерминальный, а β может быть равным ε, то есть пустому слову.

В следующих определениях $G = \{N, \Sigma, P, S\}$ – порождающая грамматика.

Опеределение: Наименьшее рефлексивное транзитивное отношение $\vdash_G$ называется *отношением выводимости* в грамматике G, если

$$\forall (\alpha \to \beta) \in P,\ \ \forall \varphi, \psi \in (N \cup \Sigma)^* : \varphi \alpha \psi \vdash_G \varphi \beta \psi$$

По сути, это операция замены левой части на правую часть несколько раз, возможно, нуль.

Опеределение: Слово $w \in \Sigma^*$ называется *выводимым* в грамматике G, если $S \vdash_G w$, то есть из стартового символа достижимо слово w.

Опеределение: Язык L *распознаётся* грамматикой G, если $L = \{w \in \Sigma^* \mid S \vdash_G w\}$, то есть язык L состоит из таких слов, которые выводимы в грамматике G. Если L распознаётся грамматикой G, то его обозначают как $L(G)$.

Праволинейные грамматики и праволинейные языки

Определение: Грамматика G называется праволинейной, если правила из P имеют вид либо $A \to wB$, либо $A \to w$, где A, B — нетерминальные символы, то есть $A, B \in N$, и $w \in \Sigma^*$.

Определение: Язык $L(G)$ называется праволинейным, если грамматика G, которой распознаётся (задаётся) язык L, является праволинейной.

Теорема о совпадении классов автоматных и праволинейных языков

Теорема: Множество автоматных языков равно множеству языков, задаваемых праволинейными грамматиками.

▲ **Грамматика $\to$ Автомат:** Состояния в автомате = нетерминалы в грамматике + сток

Пусть дан язык, задаваемый праволинейной грамматикой: $\mathcal{L} = \mathcal{L}(G),\ G = \{N, \Sigma, P, S\}$

Построим $M = \{N \cup \{q_f\}, \Sigma, \Delta', S, \{q_f\}\}$, где

$\Delta' = \{\{A, w\} \to B$, если $A \to wB \in P\} \cup \{\{A, w\} \to q_f$, если $A \to w \in P\}$

Надо показать, что $\mathcal{L}(G) = \mathcal{L}(M)$

$\underline{\mathcal{L}(G) \subset \mathcal{L}(M)}$: Индукция по длине вывода грамматики (количество $\vdash$)

$$w \in \mathcal{L}(G) \Rightarrow S \vdash_G w \Rightarrow ..?.. \Rightarrow \{S, w\} \vdash_M \{q_f, \varepsilon\} \Rightarrow w \in \mathcal{L}(M) \qquad (*)$$

Покажем, что если $C \vdash_G wD$ (за k шагов), то $\{C, w\} \vdash_M \{D, \varepsilon\}$.

Индукция по k:

$k = 0$:

$$C \vdash_G C \Rightarrow w = \varepsilon \Rightarrow \{C, \varepsilon\} \vdash_M \{C, \varepsilon\}$$

Переход:

$$C \vdash wD \text{ за } k \text{ шагов } \Rightarrow \exists E, u, v\ :\ C \vdash_G uE \text{ за } k-1 \text{ шаг },uE \vdash_G uvD \text{ за 1 шаг } (w = uv)$$

Тогда по предположению индукции

$$\{C, u\} \vdash_M \{E, \varepsilon\}, \{E, v\} \vdash_M \{D, \varepsilon\} \Rightarrow \{C, uv\} \vdash_M \{D, \varepsilon\} \text{ (по транзитивности)}$$

Если $C \vdash_G w$, то $\{C, w\} \vdash_M \{q_f, \varepsilon\}$

Так как из q_f нет переходов, то $\exists E : C \vdash_G uE$ (за $k-1$), $uE \vdash_G uv$ (за 1) $\ (E \to v \in P)$

Тогда $\{C, uv\} \vdash_M \{E, \} \vdash \{q_f, \varepsilon\}$

Чтобы доказать $(*)$, достаточно вместо C подставить S

$\{S, w\} \vdash \{q_f, \varepsilon\} \Rightarrow w \in \mathcal{L}(M)$

$\underline{\mathcal{L}(M) \subset \mathcal{L}(G)}$: $w \in \mathcal{L}(M) \Rightarrow \{S, w\} \vdash_M \{q_f, \varepsilon\}$

Покажем, что если $\{C, w\} \vdash_M \{D, \varepsilon\}$, то $C \vdash_G wD$

Индукция по длине пути в автомате:

База: $k = 0$

$$\{C, w\} \vdash_M \{D, \varepsilon\},\ \ C = D,\, w = \varepsilon \ \Rightarrow\ C \vdash_G C$$

Переход:

$$\exists E, u : \langle C, w\rangle \vdash_M \langle E, u\rangle \vdash_M \{D, \varepsilon\}$$

$$w = uv \Rightarrow \text{ по предположению индукции: } C \vdash_G uE,\ E \vdash vD \Rightarrow C \vdash_G uvD = wD$$

Так как из q_f нет переходов, то $\{S, w\} \vdash \{q_f, \varepsilon\} \Rightarrow$

$\exists E, u, v : \{S, uv\} \vdash_M \{E, v\} \vdash_M \{q_f, \varepsilon\} \Rightarrow \{S, u\} \vdash \{E, \varepsilon\} \Rightarrow S \vdash_G uE$

$$\{E, v\} \vdash_M \{q_f, \varepsilon\} \Rightarrow E \vdash_G v$$

$$S \vdash_G uE,\ E \vdash_G v \Rightarrow S \vdash_G uv$$

Автомат $\to$ Грамматика: Идея: возьмём автомат с 1 завершающим состоянием q_{stock} (и из этого q_{stock} нет переходов)

$M = \{Q, \Sigma, \Delta, q_0, \{q_{stock}\}\}$

$G = \{Q \setminus \{q_{stock}\}, \Sigma, P, q_0\}$

$$P = \{q_1 \vdash wq_2 \mid q_2 \neq q_{stock} \text{ и } \{q_1, w\} \to q_2 \in \Delta\} \cup \{q_1 \to w \mid \{q_1, w\} \to q_{stock} \in \Delta\}$$

Доказательство аналогично *Грамматика $\to$ Автомат*

11. Контекстно-свободные грамматики и языки. Примеры контекстно-свободных языков. Замкнутость и незамкнутость КС-языков относительно теоретико-множественных операций (можно пользоваться примерами не КС-языков).

Определение: *Контекстно-свободная грамматика* - грамматика, все правила которой имеют вид $A \to \alpha$, где $A \in N, \alpha \in (N \cup \Sigma)^*$. Язык называют *контекстно-свободным*, если он задается контекстно-свободной грамматикой.

Пример: КС-язык: $L = \{a^n b^m c^m\}$. Грамматика: $S \to AT, A \to aA, A \to \varepsilon, T \to bTc, T \to \varepsilon$

Пример: Не КС-язык (понадобится для доказательств): $L = \{a^n b^n c^n\}$

▲ Зафиксируем p в лемме о разрастании. Рассмотрим $w = a^p b^p c^p = xuyvz, |uv| > 0, |uyv| \leq p$. Заметим, что в uyv и uv не может быть трех разных букв из a, b, c. Не умаляя общности $|uyv|_c = 0, |uyv|_b > 0$. Рассмотрим $k = 2$:

$$|w'|_b = |xu^2yv^2z|_b = |xuyvz|_b + |uv|_b > p$$

$$|xu^2yv^2z|_c = p + |uv|_c = p$$

Следовательно, $|w'|_b \neq |w'|_c$, а значит w' не лежит в языке и выполнено отрицание леммы о разрастании, то есть язык не является КС ■

Утверждение: КС-грамматики замкнуты относительно объединения и конкатенации

▲ 1. Объединение: конструктивно построим объединение грамматик: создадим стартовую вершину S' и два правила $S' \to S_1, S' \to S_2$, где S_i - стартовый нетерминал первой или второй грамматики соответственно. Если нетерминальные символы в грамматиках совпадают переименуем их в одной из грамматик.

2. Конкатенация: аналогично, построим КС-грамматику для языка $L_1 L_2$, добавив правило $(S' \to S_1 S_2)$. ■

Утверждение: КС-грамматики не замкнуты относительно пересечения и дополнения

▲ 1. Пересечение:

$$L_1 = \{a^n b^m c^m\},$$

$$L_2 = \{a^n b^n c^m\}$$

$$L_1 \cap L_2 = \{a^n b^n c^n\} \text{ - не КС-язык}$$

2. Дополнение:

$$L = \{a^n b^n c^n\} \text{ - не КС-язык}$$

$$\overline{L} = \{\text{easy cases}\} \cup \{a^k b^l c^m | k \neq l \vee l \neq m \vee k \neq m\}$$

Оба языка из объединения - КС $\Rightarrow \overline{L}$ - КС. Но $L = \overline{\overline{L}}$ не КС $\Rightarrow$ множество КС языков не замкнуто относительно дополнения ■

12. Устранение бесполезных вспомогательных символов и eps-правил для КС-грамматик.

Определение: Символ $Y \in N$ называется порождающим, если существует слово $w \in \Sigma^*$, такое что $Y \vdash w$.

Определение: Символ $D \in N$ называется достижимым, если существуют некоторые φ, $\psi \in (N \cup \Sigma)^*$, такие что $S \vdash \varphi D \psi$.

Определение: Символ $U \in N$ называется бесполезным, если он непорождающий или недостижимый.

Определение: Символ $E \in N$ называется ε-порождающим, если $E \vdash \varepsilon$.

Утверждение: Для любой контекстно-свободной грамматики существует эквивалентная КС-грамматика без бесполезных символов.

▲ Приведём алгоритм преобразования КС-грамматики: сначала найдём непорождающие символы, удалим их из грамматики, затем найдём и удалим недостижимые символы.

Пусть G_1 — грамматика, преобразованная из G путём удаления непорождающих символов и всех правил, содержащих непорождающие символы. Покажем, почему $L(G) = L(G_1)$. $L(G_1) \subset L(G)$, так как количество правил уменьшается. Пусть $w \in L(G) \setminus L(G_1)$, тогда в дереве вывода есть непорождающий символ C:

$$S \vdash \varphi C \psi \vdash w,\ w = xyz$$

Но $\varphi \vdash x$, $C \vdash y$, $\psi \vdash z$, откуда C — порождающий символ. Противоречие.

Пусть G_2 — грамматика, преобразованная из G_1 путём удаления всех недостижимых символов и правил, содержащих их. Покажем, почему $L(G_1) = L(G_2)$. $L(G_2) \subset L(G_1)$, так как количество правил уменьшается. Пусть $w \in L(G_1) \setminus L(G_2)$, тогда существует недостижимый символ D, что $S \vdash \varphi D \psi \vdash w$, но тогда D по определению является достижимым. Противоречие.

Проверим, что не появилось новых непорождающих символов. Пусть B — непорождающий символ в G_2, значит B достижим в G_1. Тогда B — порождающий символ в G_1, то есть $B \vdash_{G_1} u$. Так как B стал непорождающим после удаления недостижимых символов, значит, что на пути вывода $B \vdash u$ был недостижимый символ C, чего не может быть так как есть путь $S \to B \to C$ - противоречие ■

Про удаление ε-порождающих символов см. билет 13

13. Нормальная форма Хомского для КС-грамматик. Алгоритм приведения к нормальной форме Хомского.

Опеределение: КС-грамматика находится в нормальной форме Хомского, если все правила имеют такой и только такой вид:

1. $A \to a\,(A \in N,\ a \in \Sigma)$;
2. $A \to BC\,(B, C \in N;\ B, C \neq S)$;
3. $S \to \varepsilon$.

Утверждение: Любую КС-грамматику можно привести к нормальной форме Хомского с помощью алгоритма, который состоит из следующих шагов:

1. Удаление непорождающих символов
2. Удаление недостижимых символов
3. Удаление смешанных правил $D \to aBc$
4. Удаление длинных правил $A \to A_1A_2A_3A_4$
5. Удаление ε-порождающих символов
6. Обработка пустого слова
7. Удаление унарных (одиночных) правил $A \to B$

1-2. После удаления непорождающих и недостижимых символов будет получена эквивалентная грамматика. Подробное доказательство этого есть в вопросе 12. Обозначим полученную грамматику за G_2.

3. Для удаления смешанных правил сделаем замену правила вида $A \to dBcEf$ на правила следующего вида: $A \to DBCEF$, $D \to d$, $C \to c$, $F \to f$.

   Обозначим полученную грамматику за G_3. Покажем, что $w \in L\,(G_2) \Leftrightarrow w \in L\,(G_3)$. В основе доказательства лежит идея, что в дереве вывода нетерминальные символы можно выводить в любом порядке.

   $\Rightarrow$

$$A \vdash_1 dBcEf \vdash dw_Bcw_Ef\ \ (G_2)$$
$$A \vdash DBCEF \vdash dBcEf \vdash dw_Bcw_Ef\ \ (G_3)$$

   $\Leftarrow$ Если в дереве вывода встречается $A \vdash DBCEF$, то раскрываем в первую очередь правила вида $D \to d$, а потом повторяем те же действия что в грамматике G_2

4. Удалим длинные правила вида $B \to A_1A_2\ldots A_n$ с помощью замены на правила следующего вида:

$$B \to A_1B_1 \qquad B_1 \to A_2B_2 \qquad B_2 \to A_3B_3 \qquad \dots \qquad B_{n-1} \to A_{n-1}A_n$$

Обозначим полученную грамматику за G_4. Покажем, что $w \in L(G_3) \Leftrightarrow w \in (G_4)$.

$\Rightarrow$ Пусть $w \in L(G_3)$. Тогда $S \vdash \varphi B\psi \vdash_1 \varphi A_1A_2\dots A_n\psi \vdash w$. Посмотрим, что происходит в G_4:

$$S \vdash \varphi B\psi \vdash \varphi A_1B_1\psi \vdash \varphi A_1A_2B_2\psi \vdash \dots \vdash \varphi A_1A_2\dots A_{n-2}B_{n-1}\psi \vdash \varphi A_1\dots A_n\psi \vdash w$$

$\Leftarrow$ Пусть $w \in L(G_4)$. Рассмотрим вывод $S \vdash w$. Возможны два варианта:

1. B_k не встречается на пути вывода. Тогда слово выводимо и в G_4, и в G_3;
2. B_k встречается на пути вывода. Тогда встречаются и все нетерминалы вида B_1, ..., B_{n-1} по построению правил. Доказательство корректности аналогично доказательству в другую сторону.

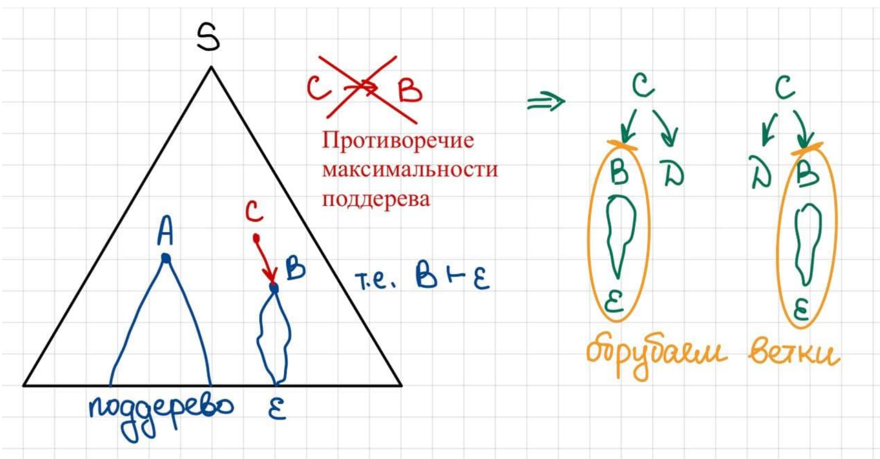
5. Теперь остались правила вида:

$$A \to a \qquad A \to BC \qquad A \to B \qquad A \to \varepsilon$$

Будем удалять ε правила:

1. Если $A \to BC$ и $C \vdash \varepsilon$, то добавим $A \to B$
2. Если $A \to BC$ и $B \vdash \varepsilon$, то добавим $A \to C$

Затем просто удаляем правила $A \to \varepsilon$. Получили грамматику G_5. Покажем корректность, то есть $w \in L(G_4) \Leftrightarrow w \in L(G_5)$ и $w \neq \varepsilon$.



▲ $\Rightarrow$ Пусть $w \in L(G_4)$. Выделим поддеревья максимальной мощности из которых выводится ε в дереве вывода w. Рассмотрим корень такого поддерева. Он не мог быть получен по правилу вида $C \to B$, так как тогда $C \vdash B \vdash \varepsilon$ и C можно было бы включить в это поддерево, что противоречило бы максимальности. Поэтому корень B был получен по правилу $C \to BD$ или $C \to DB$ (без ограничения общности рассмотрим первое). Тогда в G_4: $C \vdash_1 BD \vdash \varepsilon u = u$, а в G_5: $C \vdash_1 D \vdash u$, то есть ничего не поменялось $\Rightarrow w \in L(G_5)$

$\Leftarrow$ Пусть $w \in L(G_5)$. Рассмотрим правила вида $C \to D$ в дереве вывода w в G_5, которых не было в G_4. Значит в G_4 было одно из правил $C \to BD, C \to DB$ и $B \vdash \varepsilon \Rightarrow w \in L(G_4)$■

6. Если $S \vdash \varepsilon$, то построим грамматику G_6 следующим образом:

   1. S' — новый стартовый нетерминал;
   2. $S' \to S$ — новое правило;
   3. Если $S \vdash \varepsilon$, то $S' \to \varepsilon$ — новое правило.

   Данная грамматика будет эквивалентной. Этим шагом мы еще и частично приблизились ко второму пункту в определении НФ Хомского (не зацикливаемся на старте)

7. Рассмотрим последовательности правил:

$$B \to B_1 \to B_2 \to \cdots \to B_n \to CD$$
$$B \to B_1 \to B_2 \to \cdots \to B_n \to a$$

   Удалим унарные одиночные правила, заменим последовательности правил на $B \to CD$ и $B \to a$ соответственно (транзитивное замыкание). Обозначим полученную грамматику за G_7.

   Покажем, что $L(G_6) = L(G_7)$. (= док-ву удаления ε-переходов в автомате).

   $\Leftarrow$ Для всех правил вида $B \to a$ или $B \to CD$ верно, что $B \vdash_{G_6} a$, $B \vdash_{G_6} CD$. Никаких новых выводов добавлено не было, тем самым $L\left(G_7\right) \subseteq L\left(G_6\right)$.

   $\Rightarrow$ Покажем, что $B \vdash_{G_6} w \Rightarrow B \vdash_{G_7} w$ индукцией по длине вывода.

   База. Вывод за один шаг: $B \vdash_{G_6,1} w \implies (B \to w) \in P_{G_6}, w \in \Sigma \implies (B \to w) \in P_{G_7} \implies B \vdash_{G_7,1} w$.

   Переход. Добавленное правило имеет вид $B \to C_1 \dots C_n$, где $n \in \{1, 2\}$, $B \to C_1 \dots C_n \vdash_{G_6} w_1 \dots w_n$, по предположению индукции $C_i \vdash_{G_7} w_i$.

   Пусть $n = 2$. Тогда правило имеет вид $B \to CD$, $B \vdash_{G_7} CD \vdash_{G_7} w_1 w_2$.

   Пусть $n = 1$. Тогда вывод начинается с правила $B \to C_1$. Рассмотрим первое правило вывода, не являющееся одиночным. Тогда:

   1. $B \vdash C_1 \vdash C_2 \vdash \cdots \vdash C_m \vdash a$, $a \in \Sigma$, тогда по построению существует правило $B \to a$, $B \vdash_{G_7} a$;
   2. $B \vdash C_1 \vdash C_2 \vdash \cdots \vdash CD \vdash w_1 w_2$. По построению существует правило $B \to CD$, по предположению индукции $B \vdash_{G_7} CD \vdash w_1 w_2$.

   Построенная грамматика G_7 — грамматика в нормальной форме Хомского.

14. Алгоритм Кока-Янгера-Касами синтаксического разбора для КС-грамматик.

Алгоритм Кока-Янгера-Касами принимает на вход грамматику G в нормальной форме Хомского и слово w. В качестве выхода выдаётся информация, принадлежит ли слово w языку L, который задаётся грамматикой G.

Идея алгоритма

Пусть $G = \langle N, \Sigma, P, S \rangle$ и $|w| = n$.

Будем решать задачу динамическим программированием. Заведем трехмерный массив d размером $|N| \times n \times n$, состоящий из логических значений, и $d[A][i][j] = true$ тогда и только тогда, когда из нетерминала А правилами грамматики можно вывести подстроку $w[i \dots j]$.

Рассмотрим все пары $\{\langle j, i \rangle \mid j - i = m\}$, где m – константа и $m < n$.

1) $i = j$. Инициализируем массив для всех нетерминалов, из которых выводится какой-либо символ строки w. В таком случае $d[A][i][i] = true$, если в грамматике G присутствует правило $A \to w[i]$. Иначе $d[A][i][i] = false$.

2) $i \neq j$. Значения для всех нетерминалов и пар $\{\langle j', i' \rangle \mid j' - i' < m\}$ уже вычислены, поэтому

$$d[A][i][j] = \bigvee_{A \to BC} \bigvee_{k=i}^{j-1} d[B][i][k] \wedge d[C][k+1][j]$$

То есть, подстроку $w[i \dots j]$ можно вывести из нетерминала A, если существует продукция вида $A \to BC$ и такое k, что подстрока $w[i \dots k]$ выводима из B, а подстрока $w[k+1 \dots j]$ выводится из C. Ответом будет значение $d[S][1][n]$.

Доказательство корректности

Проведём индукцию по длине слова.

База. Пусть $j = i$. Тогда $dp[A][i][j] = True$ появилась на этапе инициализации, что по построению равносильно тому, что правило $(A \to w[i]) \in P$, что эквивалентно тому, что $A \vdash w[i : j]$, $j = i$, то есть $A \vdash w[i]$.

Переход. Пусть $dp[A][i][j] = True$. Тогда по построению и предположению индукции существуют число k и нетерминальные символы B и C, такие что $A \vdash_1 BC$, $B \vdash w[i : k]$, $C \vdash w[k+1 : j]$, откуда $dp[B][i][k] = dp[C][k+1][j] = True$, так как $A \vdash_1 BC$, то $(A \to BC) \in P$, $A \vdash w[i : j]$.

Асимптотика

Обработка правил вида $A \to w[i]$ выполняется за $O(n \cdot |P|)$.

Проход по всем подстрокам выполняется за $O(n^2)$. В обработке одной подстроки присутствует цикл по всем правилам вывода и по всем разбиениям на две подстроки, следовательно обработка работает за $O(n \cdot |P|)$. В итоге получаем конечную сложность $O(n^3 \cdot |P|)$.

15. Лемма о разрастании для КС-языков. Примеры языков, не являющихся КС-языками.

Лемма о разрастании для КС-языков

Лемма: Пусть L – КС-язык. Тогда существует p, такое что для любого слова $w \in L$, длина которого не меньше, чем p, существуют такие слова x, u, y, v, z, принадлежащие Σ^*, что $w = xuyvz$, $|uv| > 0$, $|uyv| \le p$, что для любого $k \in \mathbb{N}$ выполняется, что $xu^k yv^k z \in L$.

Кванторная версия:

$$\exists p : \forall w \in L : |w| \ge p : \exists x,\ u,\ y,\ v,\ z \in \Sigma^* : w = xuyvz,\ |uv| > 0,\ |uyv| \le p : \forall k \in \mathbb{N} : xu^k yv^k z \in L$$

▲ Рассмотрим грамматику G в нормальной форме Хомского: $L = L(G)$. Выберем $p = 2^{|N|}$, где $|N|$ — количество нетерминальных символов. Тогда $|w| \ge p = 2^{|N|}$. Дерево вывода является бинарным деревом, и тогда существует «ветвь» дерева вывода уровня хотя бы $|N|$ (уровни в 0 индексации). Рассмотрим «ветвь» максимальной глубины: в ней количество нетерминалов будет хотя бы $|N| + 1$. Тогда по принципу Дирихле, существует нетерминал A, такой что $S \vdash xAz \vdash xuAvz \vdash xuyvz$ и $A \vdash uAv$, который повторяется не менее двух раз. Среди всех возможных нетерминалов A выберем тот, который находится ниже всех, то есть его глубина относительно корня наибольшая.

$$\begin{tikzcd} & \mathrm{S} \arrow[dl] \arrow[d] \arrow[dr] & \\ \mathrm{x} & \mathrm{A} \arrow[dl] \arrow[d] \arrow[dr] & \mathrm{z} \\ \mathrm{u} & \mathrm{A} \arrow[d] & \mathrm{v} \\ & \mathrm{y} & \end{tikzcd}$$

Покажем, что $|uyv| \le p = 2^{|N|}$. Пусть $|uyv| > p = 2^{|N|}$, тогда для дерева со стартом в A можно сделать те же самые операции, значит, существует уровень, который больше $|N|$, значит, существует в поддереве пара $B \vdash B$, и A – не самый глубокий нетерминал.

Покажем, что $|uv| > 0$. Для любого нетерминального символа $C \in N$ верно, что C не является ε-порождающим. Рассмотрим D такой, что $D \vdash KA$ (за один шаг), $K \vdash r$, где r — суффикс u, $|r| > 0$. Отсюда следует, что $|uv| \ge |u| \ge |r| > 0$. ■

Пример: Не КС-язык: $L = \{a^n b^n c^n\}$

▲ Зафиксируем p в лемме о разрастании. Рассмотрим $w = a^p b^p c^p = xuyvz, |uv| > 0, |uyv| \le p$. Заметим, что в uyv и uv не может быть трех разных букв из a, b, c. Не умаляя общности $|uyv|_c = 0, |uyv|_b > 0$. Рассмотрим $k = 2$:

$$|w'|_b = |xu^2yv^2z|_b = |xuyvz|_b + |uv|_b > p$$

$$|xu^2yv^2z|_c = p + |uv|_c = p$$

Следовательно, $|w'|_b \ne |w'|_c$, а значит w' не лежит в языке и выполнено отрицание леммы о разрастании, то есть язык не является КС ■

16. Автоматы с магазинной памятью (МП-автоматы). Различные варианты определений. Языки, распознаваемые МП-автоматами.

Определение: Автомат с магазинной памятью (МП-автомат) — кортеж $M = \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle$, где:

1. Q — множество состояний, Q — конечное множество, то есть $|Q| < \infty$;
2. Σ — алфавит, $|\Sigma| < \infty$;
3. Γ — стековый алфавит, $|\Gamma| < \infty$, $\Gamma \cap \Sigma = \varnothing$;
4. $\Delta \subset (Q \times \Sigma^* \times \Gamma^*) \times (Q \times \Gamma^*)$ — множество переходов, $|\Delta| < \infty$;
5. $q_0 \in Q$ — стартовое состояние;
6. $F \subset Q$ — множество завершающих состояний.

Переходы имеют вид $\langle q_1, w, \alpha \rangle \to \langle a_2, \beta \rangle$, $w \in \Sigma^*$, $\alpha \in \Gamma^*$, *то есть когда находимся в состоянии q_1, снимаем со стека слово α, стек растёт слева направо, читаем слово w, переходим в состояние q_2, добавляем на стек слово β.*

Определение: Конфигурация МП-автомата M – кортеж $\langle q, u, \gamma \rangle$, где $q \in Q$, $u \in \Sigma^*$, $\gamma \in \Gamma^*$.

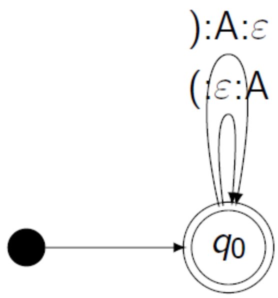
Определение: Отношение выводимости $\vdash$ — наименьшее рефлексивное транзитивное отношение, что для любого перехода $(\langle q_1, u, \alpha \rangle \to \langle q_2, \beta \rangle) \in \Delta$ выполнено следующее:

$$\forall v \in \Sigma^*, \eta \in \Gamma^* : \langle q_1, uv, \eta\alpha \rangle \vdash \langle q_2, v, \eta\beta \rangle$$

Языки, распознаваемые МП-автоматами

Определение: Пусть M — МП-автомат, язык $L(M)$, распознаваемый МП-автоматом M — множество $\{w \in \Sigma^* | \exists q \in F : \langle q_0, w, \varepsilon \rangle \vdash \langle q, \varepsilon, \varepsilon \rangle\}$.

Пример: Язык правильных скобочных последовательностей распознаётся следующим МП-автоматом:

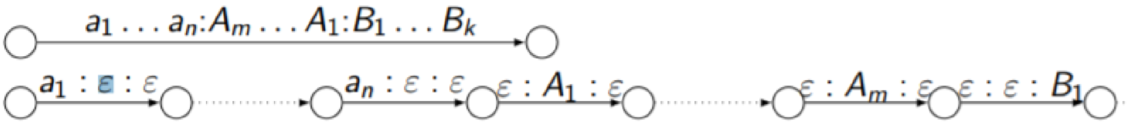


Упрощения МП-автоматов

Утверждение: Для любого МП-автомата существует эквивалентный МП-автомат, для которого выполнено соотношение:

$$\forall\, (\langle q_1, u, \alpha\rangle \to \langle q_2, \beta\rangle) \in \Delta : |u| \leqslant 1, |\alpha| + |\beta| \leqslant 1$$

▲

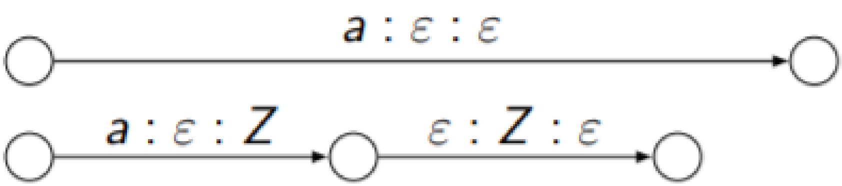


■

Утверждение: Для любого МП-автомата существует эквивалентный МП-автомат, для которого выполнено соотношение:

$$\forall\, (\langle q_1, u, \alpha\rangle \to \langle q_2, \beta\rangle) \in \Delta : |u| \leqslant 1, |\alpha| + |\beta| = 1$$

▲



■

17. Совпадение классов КС-языков и языков, распознаваемых МП-автоматами: построение автомата по грамматике.

В вопросах 16 и 17 предлагается доказать в одну из сторон следующую теорему:

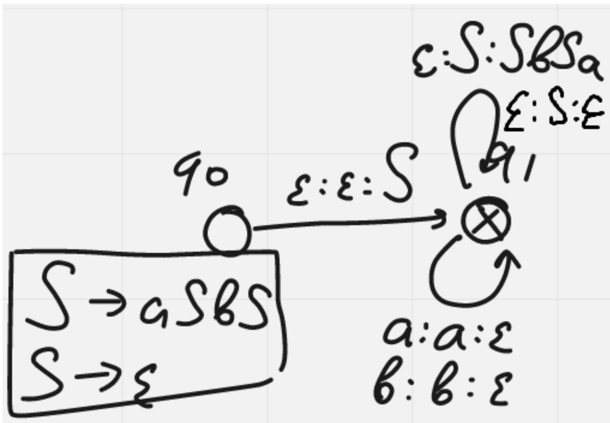
Теорема: Язык L является МП-автоматным тогда и только тогда, когда L является контекстно-свободным.

Gram $\Longrightarrow$ Automaton

▲ Идея: снимаем со стека левую часть правила, добавляем правую часть правила

Рассмотрим контекстно-свободную грамматику $G = \langle N, \Sigma, P, S\rangle$. Автомат строим следующим образом: q_0 — стартовое (начальное) состояние, q_1 — единственное завершающее состояние, $Q = \{q_0, q_1\}$. Переходы из q_1 в q_1 имеют вид либо $\langle q_1, a, a\rangle \to \langle q_1, \varepsilon\rangle$, если a — некоторый терминальный символ, либо $\langle q_1, \varepsilon, S\rangle \to \langle q_1, SbSa\rangle$, если существует правило вида $S \to aSbS$, где $S \in N$. *Заметим, что при обработке правил, где левая часть — некоторый нетерминал, мы добавляем в стек* ***развёрнутую правую часть*** *правила.*

Чтобы было видно, что происходит, приведём пример. Пусть правила КС-грамматики G следующие: $(S \to \varepsilon), (S \to aSbS)$. Тогда МП-автомат по КС-грамматике будет таким:



Докажем следующую лемму:

Лемма: $A \vdash w \Longleftrightarrow \langle q_1, w, A\rangle \vdash \langle q_1, \varepsilon, \varepsilon\rangle$

Докажем индукцией по длине дерева вывода (количеству рёбер в дереве). Считаем её равной k.

База. $k = 1$, $A \vdash_1 w$. Пусть $w = w_1 w_2 \dots w_n$. Так как $A \to w_1 \dots w_n$, то $\langle q_1, \varepsilon, A\rangle \to \langle q_1, w^R\rangle$. Значит:

$$\langle q_1, w, A\rangle \vdash \langle q_1, w, w_n w_{n-1} \dots w_1\rangle \vdash$$
$$\vdash \langle q_1, w_2 \dots w_n, w_n w_{n-1} \dots w_2\rangle \vdash$$
$$\vdash \langle q_1, w_3 \dots w_n, w_n \dots w_3\rangle \vdash \dots$$
$$\vdash \langle q_1, w_n, w_n\rangle \vdash \langle q_1, \varepsilon, \varepsilon\rangle$$

Переход. Посмотрим на первое раскрытие: $A \vdash_1 \alpha \vdash w$. Здесь $\alpha = \alpha_1 \dots \alpha_n \vdash w_1 \dots w_n = w$, $\alpha_i \in N \cup \Sigma$. Тогда $\langle q_1, w, A\rangle \vdash \langle q_1, w, \alpha_n \dots \alpha_1\rangle$.

Если $\alpha_n \in N$, то тогда $\alpha_n \vdash w_n$, и по предположению индукции $\langle q_1, w_n, \alpha_n \rangle \vdash \langle q_1, \varepsilon, \varepsilon \rangle$.

Если $\alpha_n \in \Sigma$, то $\alpha_n \vdash w_n$, $\alpha_n = w_n$, и тогда $\langle q_1, w_n, w_n \rangle \vdash \langle q_1, \varepsilon, \varepsilon \rangle$, так как это правило грамматики.

В итоге:

$$\langle q_1, w, A \rangle \vdash \langle q_1, w, \alpha_n \dots \alpha_1 \rangle \vdash \langle q_1, w_1 \dots w_n, \alpha_n \dots \alpha_1 \rangle \vdash \langle q_1, w_2 \dots w_n, \alpha_n \dots \alpha_2 \rangle \vdash \dots \vdash \\ \langle q_1, w_n, \alpha_n \rangle \vdash \langle q_1, \varepsilon, \varepsilon \rangle.$$

Теперь нужно показать в обратную сторону: если $\langle q_1, w, A \rangle \vdash \langle q_1, \varepsilon, \varepsilon \rangle$, то $A \vdash w$. Это сделаем с помощью индукции по количеству переходов.

База. Пусть всего один переход, тогда $A \in \Sigma$, и $\langle q_1, w, A \rangle \vdash_1 \langle q_1, \varepsilon, \varepsilon \rangle$, откуда $w = A$, и $A \vdash w$, так как $\vdash$ обладает свойством рефлексивности.

Переход. $\langle q_1, w, A \rangle \vdash_k \langle q_1, \varepsilon, \varepsilon \rangle$, где $k > 1$. Тогда $A \in N$, откуда если $A \to \alpha_1 \dots \alpha_n$, где $\alpha_m \in (N \cup \Sigma)$, то:

$$\langle q_1, w, A \rangle \vdash_1 \langle q_1, w, \alpha_n \dots \alpha_1 \rangle \vdash \langle q_1, \varepsilon, \varepsilon \rangle$$

Замечание: За один шаг мы снимаем ровно один элемент со стека:

$$\begin{gathered} \alpha_1 \to \beta_1 \\ \alpha_2 \to \beta_2 \\ \vdots \\ \alpha_n \to \beta_n \end{gathered}$$

Пусть $\alpha_1 \to \beta_1$, $\alpha_1 \in N$, $\langle q_1, w, \alpha_n \dots \alpha_1 \rangle \vdash_1 \langle q_1, w, \alpha_n \dots \alpha_2 \beta_1^R \rangle$. Дождёмся, пока на стеке останется $\alpha_n \dots \alpha_2$: так как в конце стек пустой и мы считаем ровно 1 символ. Тогда в этом моменте:

$$\langle q_1, w, \alpha_n \dots \alpha_1 \rangle \vdash \langle q_1, w', \alpha_n \dots \alpha_2 \rangle \Longrightarrow \exists w_1 : w = w_1 w' : \langle q_1, w_1, \alpha_1 \rangle \vdash \langle q_1, \varepsilon, \varepsilon \rangle \overset{hypothesis}{\Longrightarrow} \alpha_1 \vdash w_1$$

По аналогии, $\alpha_m \vdash w_m$ для любого $m = \overline{1, n}$.

Итого имеем, что $A \to \alpha_1 \dots \alpha_n$, $\alpha_m \vdash w_m$, $w_1 \dots w_n = w$. Из всего этого следует, что $A \vdash w_1 \dots w_n = w$. Переход доказан.

Из доказанной леммы следует, что:

$$w \in L(G) \Longleftrightarrow S \vdash w \Longleftrightarrow \langle q_1, w, S \rangle \vdash \langle q_1, w, S \rangle \vdash \langle q_1, \varepsilon, \varepsilon \rangle \text{ (по лемме)}$$
$$w \in L(M) \Longleftrightarrow \langle q_0, w, \varepsilon \rangle \vdash_1 \langle q_1, w, S \rangle \vdash \langle q_1, \varepsilon, \varepsilon \rangle$$

■

18. Совпадение классов КС-языков и языков, распознаваемых МП-автоматами: построение грамматики по автомату.

Automaton $\Longrightarrow$ Gram

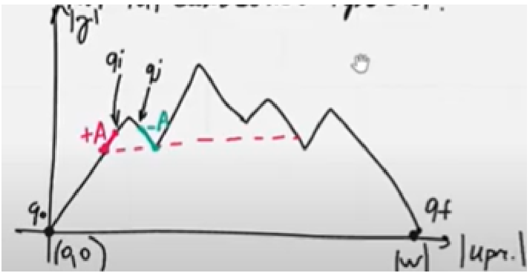
▲ Пусть $w \in L(M)$, M — МП-автомат, для которого выполнено соотношение:

$$\forall\left(\langle q_1, u, \alpha\rangle \to \langle q_2, \beta\rangle\right) \in \Delta : |u| \leqslant 1, |\alpha| + |\beta| = 1$$

Для вывода построим график «длина стека» от префикса w:

$$\langle q_0, w, \varepsilon\rangle \vdash \langle q, u, \gamma\rangle$$

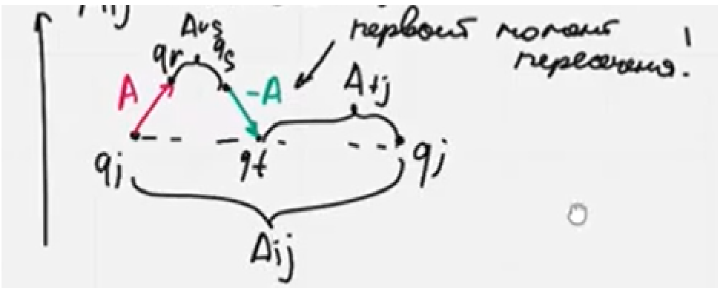
На графике — точка $(|\gamma|, |w| - |u|)$.



Далее зададим грамматику $G = \langle N, \Sigma, P, S\rangle$, где:

$N = S \cup \{A_{ij} | q_i, q_j \in Q\}$ *Под A_{ij} подразумевается то, что выводится на пути между q_i и q_j без изменения стека (нет pop()).* P — объединение следующих множеств:

1. $\{A_{ii} \to \varepsilon | q_i \in Q\}$
2. $\{S \to A_{0j} | q_j \in F\}$
3. $\{A_{ij} \to aA_{rs}bA_{tj} \; | \alpha \wedge \beta\}$, где:
   (a) α: условие, что $\langle q_i, a, \varepsilon\rangle \to \langle q_r, A\rangle \in \Delta$;
   (b) β: условие, что $\langle q_s, b, A\rangle \to \langle q_t, \varepsilon\rangle \in \Delta$.



Теперь нужно доказать следующую лемму:

Лемма: $A_{ij} \vdash_G w \Longleftrightarrow \langle q_i, w, \varepsilon\rangle \vdash_M \langle q_j, \varepsilon, \varepsilon\rangle$

$\Longrightarrow$ Докажем индукцией по длине вывода в грамматике.

База. Вывод за один шаг. $A_{ij} \to \varepsilon$. Тогда $i = j$, $w = \varepsilon$, $\langle q_i, \varepsilon, \varepsilon \rangle \vdash \langle q_i, \varepsilon, \varepsilon \rangle$.

Переход. Пусть $A_{ij} \vdash w$ за k шагов. Тогда $A_{ij} \vdash aA_{rs}bA_{tj}$, при этом:

$$\langle q_i, a, \varepsilon \rangle \to \langle a_r, A \rangle \ (1)$$
$$\langle q_s, b, A \rangle \to \langle q_t, \varepsilon \rangle \ (3)$$

Слово w имеет вид $aubv$. Тогда, используя предположение индукции, можем получить:

$$A_{rs} \vdash u \Longrightarrow \langle q_r, u, \varepsilon \rangle \overset{(2)}{\vdash} \langle q_s, \varepsilon, \varepsilon \rangle$$
$$A_{tj} \vdash v \Longrightarrow \langle q_t, v, \varepsilon \rangle \overset{(4)}{\vdash} \langle q_j, \varepsilon, \varepsilon \rangle$$

Тогда $\langle q_i, aubv, \varepsilon \rangle \overset{(1)}{\vdash} \langle q_r, ubv, A \rangle \overset{(2)}{\vdash} \langle q_s, bv, A \rangle \overset{(3)}{\vdash} \langle q_t, v, \varepsilon \rangle \overset{(4)}{\vdash} \langle q_j, \varepsilon, \varepsilon \rangle$.

$\Longleftarrow$ Проведём индукцию по количеству переходов k, которые необходимы для того, чтобы $\langle q_i, w, \varepsilon \rangle \vdash \langle q_j, \varepsilon, \varepsilon \rangle$.

База. $k = 0$. Тогда $\langle q_i, w, \varepsilon \rangle \vdash_0 \langle q_j, \varepsilon, \varepsilon \rangle$, откуда $w = \varepsilon$ и $q_i = q_j$, $A_{ij} = A_{ii}$, так как есть правило $A_{ii} \to \varepsilon$, то $A_{ii} \vdash \varepsilon$.

Переход. $\langle q_i, w, \varepsilon \rangle \vdash_k \langle q_j, \varepsilon, \varepsilon \rangle$. Так как стек пустой, то:

$$\langle q_i, w, \varepsilon \rangle \vdash_1 \langle q_r, u, A \rangle$$

A существует, так как мы либо кладём, либо снимаем со стека. Пусть $q_s \to q_t$ — это момент, когда A снят со стека. Тогда:

$$\langle q_s, v, A \rangle \vdash_1 \langle q_t, x, \varepsilon \rangle \vdash \langle q_j, \varepsilon, \varepsilon \rangle$$
$$\langle q_i, a, \varepsilon \rangle \to \langle q_r, A \rangle \in \Delta : w = au$$
$$\exists u' : u = u'v,\ \langle q_r, u', \varepsilon \rangle \vdash \langle q_s . \varepsilon, \varepsilon \rangle$$

Пользуясь предположением индукции, получаем:

$$A_{rs} \vdash u'$$
$$\langle q_s, b, A \rangle \vdash \langle q_t, \varepsilon \rangle \in \Delta : v = bx$$
$$\langle q_t, x, \varepsilon \rangle \vdash \langle q_j, \varepsilon, \varepsilon \rangle \Longrightarrow A_{tj} \vdash x$$

Так как правило $A_{ij} \to aA_{rs}bA_{tj} \in P$, то $A_{ij} \vdash aA_{rs}bA_{tj} \vdash au'bx = au'v = au = w$. Предположение доказано.

Из леммы будет следовать теорема следующим образом:

$$w \in L(G) \Longleftrightarrow S \vdash w \Longleftrightarrow \exists A_{0j} \, (q_j \in F) : S \vdash_1 A_{0j} \vdash w \Longleftrightarrow \ \text{(по лемме)}$$
$$\exists A_{0j} \, (q_j \in F) : \langle q_0, w, \varepsilon \rangle \vdash \langle q_j, \varepsilon, \varepsilon \rangle \Longleftrightarrow \exists q_j \in F : \langle q_0, w, \varepsilon \rangle \vdash \langle q_j, \varepsilon, \varepsilon \rangle \Longleftrightarrow w \in L(M)$$

□

19. Нормальная форма Грейбах для КС-грамматик. Модифицированный алгоритм Кока-Янгера-Касами для нормальной формы Грейбах: достоинства и недостатки.

Опеределение: Грамматика в нормальной форме Грейбах (grammar in Greibach normal form) — контекстно-свободная грамматика $\langle N, \Sigma, P, S\rangle$, в которой каждое правило имеет один из следующих четырёх видов:

1) $S \to \varepsilon$

2) $A \to a$

3) $A \to aB$

4) $A \to aBC$

причём $B, C \in N$, $B, C \neq S$, $a \in \Sigma$

Модифицированный алгоритм Кока-Янгера-Касами

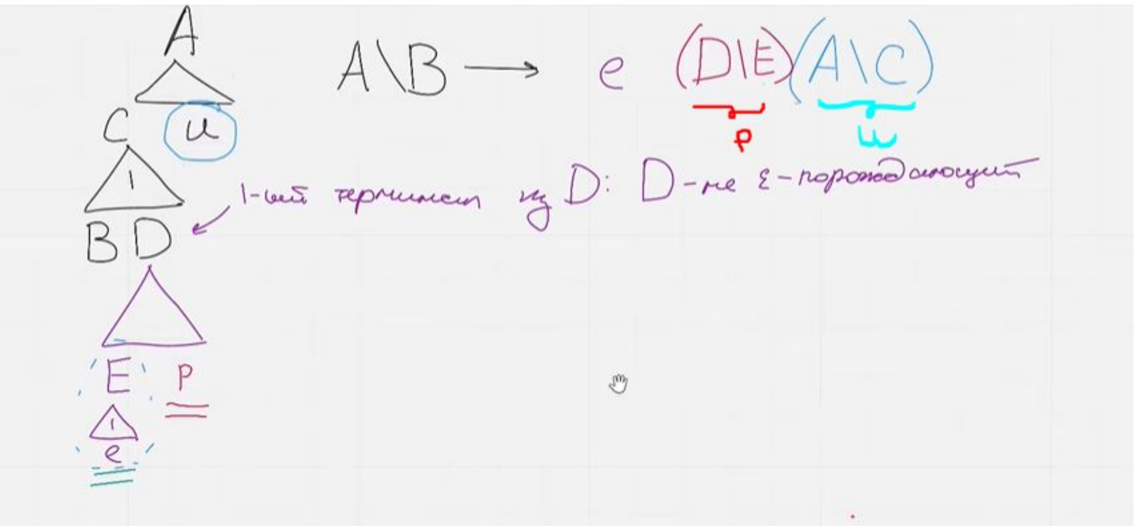
Утверждение: проще парсить из нормальной формы Грейбаха, т.к. каждый раз мы вытаскиваем одну букву $\Rightarrow$ в алгоритме Кока-Янгера-Касами константа будет ниже;

Минус: большое количество нетерминалов.

Теорема Каждая контекстно-свободная грамматика эквивалентна некоторой грамматике в нормальной форме Грейбах. (на хор 6 и выше по версии Виталия)

▲ 0. Возьмём G в НФ Хомского. Введём обозначение: $A\backslash B \vdash w \Leftrightarrow A \vdash Bw$

1. Заметим следующее:



2. Вводим G' $= \langle N', \Sigma, P', S\rangle$, где $N' = \{S\} \cup \{(A\backslash B) | A, B \in N\}$,

a $P' = \{(A\backslash A) \to \varepsilon | A \in N\} \cup \{S \to a(S\backslash A) | A \to a \in P\} \cup \{(A\backslash B) \to e(D\backslash E)(A\backslash C) | C \to BD, E \to e \in P\} \cup \{S \to \varepsilon | S \to \varepsilon \in P\}$. Отсюда $O(N') = O(N^2)$ ■

Осталось доказать: $\forall A, B \in N\ \forall w \in \Sigma^*\ A \vdash_G Bw \Leftrightarrow (A\backslash B) \vdash_{G'} w$

$\Rightarrow$: индукция по длине вывода в G. База: A = B, $w = \varepsilon$. Переход: см. картинку

$\Leftarrow$: индукция по длине вывода в G'. База: $A\backslash B \vdash_1 w \Rightarrow w = \varepsilon, A = B \Rightarrow A \vdash A\varepsilon$.

Переход: $A\backslash B \vdash_1 e(C\backslash D)(A\backslash F)$; $(C\backslash D) \vdash u$; по предположению индукции, $C \vdash Du$. Аналогично, $A \vdash Fv$. + в P есть правила $D \to e$ и $F \to BC$.

Тогда $A \vdash Fv \vdash BCv \vdash BDuv \vdash Beuv$. Почти победа!

L(G) = L(G'). $w \in L(G) \Leftrightarrow S \vdash w$.

$L(G) \subset L(G')$: пусть $S \vdash w$

1) $w = \varepsilon$, переносим правило $S \to \varepsilon$ в G'

2) $w \neq \varepsilon$. Тогда $w = au$. $S \vdash Au \vdash_1 au$, т.к. НФ Хомского. По лемме $S\backslash A \vdash u \Longrightarrow S \vdash_{G'} a(S\backslash A) \vdash au = w$.

$L(G') \subset L(G)$: $w = au$, $S \vdash_{G'} a(S\backslash A)$, где $A \to a \in P$, тогда $S\backslash A \vdash u$, в итоге $S \vdash Au = au = w$.

20. Линейные и полулинейные множества. Лемма о том, что для всякого полулинейного множества есть регулярный язык. Теорема Парика: существование КС-языка для полулинейного множества.

$w \in L : L \subset \Sigma^*; \Sigma = \{a_1, a_2, \dots, a_n\} \Rightarrow \exists \psi : \forall w \to (|w|_{a_1}, |w|_{a_2}, \dots, |w|_{a_n})$

Если L - регулярный язык, то $\psi(L)$ - ?

Введём операции: сложение - $X + Y = \{x + y | x \in X, y \in Y\}$, выпуклая оболочка: $< X >= \{\alpha_1 x_1 + \dots + \alpha_n x_n | x_1, \dots, x_n \in X, \alpha_1, \dots, \alpha_n \in \mathbb{Z}^+, \alpha_i \geqslant 0\}$

$X \subseteq \mathbb{N}^k$, $X = X_1 + < X_2 >, |X_1|, |X_2| < \infty$ - **линейное**

Примеры:

$\psi(abaab) = (3, 2)$; $\psi((ab)^*) = \{(n, n) | n \geqslant 0\}, \psi((ab)^*) =< (1, 1) >$, $\psi((ab)^*) = (0, 0)+ < (1, 1) >$ (первая компонента - базовое положение, вторая - разрастание, т.е. идейно это похоже на лемму о разрастании)

$X \subseteq \mathbb{N}^k$ - **полулинейное**, если X представимо в виде конечного объединения линейных множеств.

Соответственно, язык L - (полу)линейный, если $\psi(L)$ - (полу)линейный.

Утверждение: L - регулярный $\Rightarrow$ L - полулинейный.

Лемма: $\forall X \subseteq \mathbb{N}^{|\Sigma|}$ полулинейного $\exists$ R (полулинейный) - регулярный язык ($\psi(R) = X$)

Теорема Па'рика: L - КС язык $\Rightarrow$ L - полулинейный.

Док-во леммы:

X - полулинейное. Надо найти $R : \psi(R) = X$

$X = X_1 \cup \dots X_l$, X_i - линейное. Для X_i найдём R_i: $\psi(R_i) = X_i$; $X_i = \{x_1, \dots, x_m\}+ < \{y_1, \dots, y_t\} >$, $x_i, y_i \in \mathbb{N}^{|\Sigma|}$: $x_j \to u_j : \psi(u_j) = x_j = \{x_j^1, \dots, x_j^{|\Sigma|}\}$

$x_j \to u_j : \psi(u_j) = x_j = \{x_j^1 \dots, x_j^{|\Sigma|}\}$

$u_j = a_1^{x_j^1} \cdot a_2^{x_j^2} \cdot \ldots \cdot a_{|\Sigma|}^{x_j^{|\Sigma|}}$

$y_j \to v_j : v_j = a_1^{y^1} \cdot \ldots \cdot a_{|\Sigma|}^{y_j^{|\Sigma|}}$

$\psi((u_1 + u_2 + \dots + u_m)) = \{x_1, \dots, x_m\}$

$\psi((u_1 + u_2 + \dots + u_m)(v_1 + \dots + v_t)^*] = X_j$

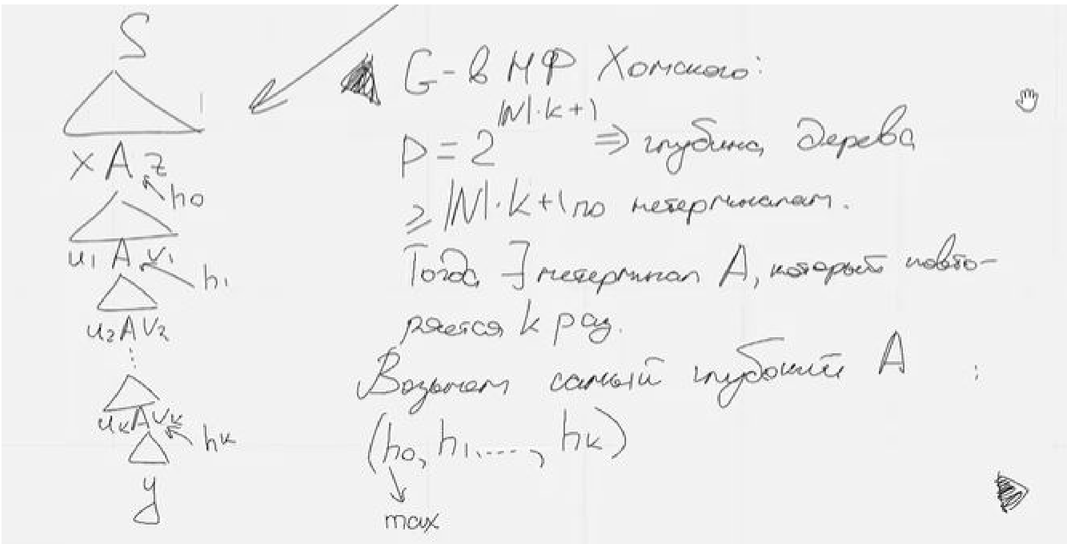
Нашли для X_i, а дальше просто объединение - вот и получили.

21. Линейные и полулинейные множества. Теорема Парика: полулинейность КС-языков.

Лемма: $\forall X \subseteq \mathbb{N}^{|\Sigma|}$ полулинейного $\exists$ R (полулинейный) - регулярный язык ($\psi(R) = X$)

Теорема Па'рика: L - КС язык $\Rightarrow$ L - полулинейный.

Введём **лемму о разрастании**: G - КС язык. Тогда: $\forall k \exists p : \forall w \in L : |w| \geqslant p \; \exists w = xu_1 \dots u_k y v_k \dots v_2 v_1 z$: $|u_i v_i| > 0, |u_1 \dots u_k y v_k \dots v_1| \leqslant p$



▲ G - в НФ Хомского, $p = 2^{|N|k+1} \Rightarrow$ глубина дерева $\geqslant |N|k + 1$ по нетерминалам. Тогда $\exists$ нетерминал A, который повторяется k раз. Возьмём самый глубокий A: $(h_0, h_1, \dots, h_k)$ ■

Док-во теоремы Парика:

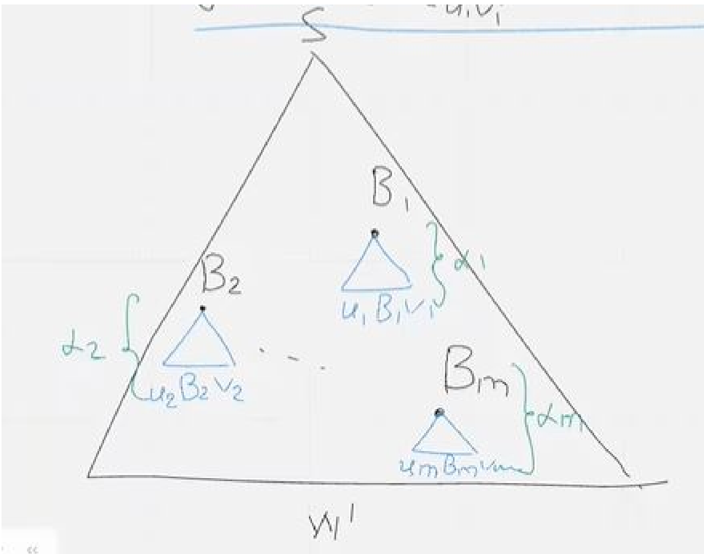
▲ $L = L(G)$: G - в НФ Хомского, $=< N, \Sigma, P, S >$. $\forall M \subset N : L_M(G) = \{w | S \vdash w$ при помощи только нетерминалов из M, причём всех $\}$.

Надо доказать, что $L_M(G)$ - полулинейно (тогда $L(G)$ - объединение конечного числа полулинейных множеств и как следствие полулинейно). Возьмём p из леммы о разрастании, $k = |N|$. $X = \{\psi(w) | w \in L_M(G), |w| \leqslant p\}$, $Y = \{\psi(uv) | |uv| \leqslant p \&\& \exists B \vdash uBv$ из нетерминалов M $\}$

Лемма: $\psi(L_M(G)) = X + < Y >$.

$\Leftarrow$: $x \in X + < Y >$. $x = x_1 + \alpha_1 y_1 + \dots + \alpha_m y_m$; $x_1 \in X \Rightarrow \exists w' : w' \in L_M(G), |w'| \leqslant p$.

$y_i \in Y \Rightarrow \exists w_i = u_i v_i$: вывод $B_i \vdash u_i B v_i$ из M. Финт ушами: $w' \in L_M(G) \Rightarrow \exists$ вывод со всеми нетерминлами B: рисуем ёлку вида



w” - слово после "подвешивания"$B_i \vdash u_i B_i v_i$ к дереву вывода; $w'' \in (L_M(G) : \psi(w'') = x$

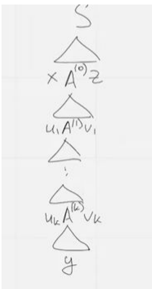
Победа!

$\Rightarrow$: $w \in L_M(G) \Rightarrow \psi(w) \in X+ < Y >$.

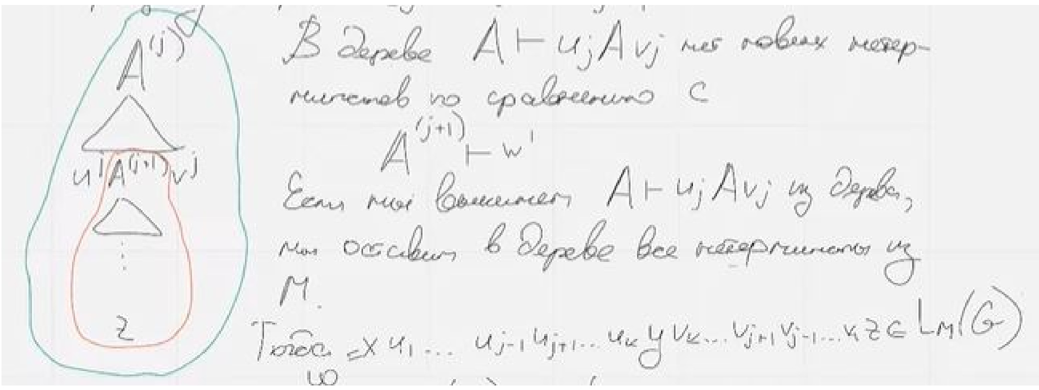
Индукция по длине слова.

База: $|w| \leqslant p \Rightarrow \psi(w) \in X$.

Переход: для $k = |N|$ лемма о разрастании:



M_i - множество нетерминалов, не считая A, которые встречаются в выводе: $A^{(i)} \vdash u_{i+1}u_{i+2}\dots u_k y v_k \dots$ Тогда $|M| > |M_0| \geqslant |M_1| \geqslant |M_2| \geqslant \dots \geqslant |M_k| \geqslant 0$; k+1 число $= |N| + 1$ число от 0 до $|M| - 1$. Тогда по принципу Дирихле $\exists j : M_j = M_{j+1}$.



$\psi(w) = \psi(\omega) + \psi(u_j v_j)$, $\psi(\omega) \in X+ < Y >$, $\psi(u_j v_j) \in Y$ (первое по предположению индукции). Ура!

■

3. Парсеры

Основная информация об алгоритме Эрли

Пусть $w \in \Sigma^*$ — слово на входе. На вход подаётся контекстно-свободная грамматика $G = \langle N, \Sigma, P, S \rangle$.

Определение: Ситуация — объект вида $(A \to \alpha \cdot \beta, i)$, где правило $(A \to \alpha\beta) \in P$, $\cdot$ — вспомогательный символ, который не принадлежит ни Σ, ни N, $i \in [0; |w|]$.

Определение: D_j — множество ситуаций вида $(A \to \alpha \cdot \beta, i)$ таких, что $\alpha \vdash w[i : j]$.

Замечание: Вывод считаем левосторонним: пусть правило $(A \to \beta) \in P$, тогда:

$$S \vdash \varphi A \psi \vdash_1 \varphi \beta \psi \Longrightarrow \varphi \in \Sigma^*$$

Замечание: Ситуация $(A \to \alpha \cdot \beta, i) \in D_j$ означает, что:

1. $S \vdash w[0 : i]$, где S — стартовый нетерминальный символ;
2. $(A \to \alpha\beta) \in P$;
3. $\alpha \vdash w[i : j]$.

Если вдруг с алгоритмом Эрли вы встречаетесь впервые, то представьте, что точка играет роль курсора, слева от которого то, что уже было введено, а справа находится то, что предстоит обработать.

Замечание: Для удобства вводится новый стартовый нетерминальный символ S', а также в грамматику G добавляется правило $(S' \to S)$. На выводимость слова это не влияет.

Операции

Всего в алгоритме Эрли поддерживаются три операции: **Scan**, **Predict**, **Complete**. *Проще говоря, Scan отвечает за «прочтение» нового символа слова, то есть появляются ситуации, которые соответствуют тому, как префикс слова мог быть выведен, Predict отвечает за генерацию возможных ситуаций при прочтении следующего символа, который является нетерминальным, то есть как бы «предсказывает», по каким правилам слово может быть выведено дальше, Complete отвечает как бы за проверку того, было ли правильным «предсказание» со стороны операции Predict.* Теперь переходим к формальным определениям операций:

$$\text{Scan: } \begin{cases} (A \to \alpha \cdot a\beta, i) \in D_j \\ w[j] = a \end{cases} \Longrightarrow (A \to \alpha a \cdot \beta, i) \in D_{j+1}$$

$$\text{Predict: } \begin{cases} (A \to \alpha \cdot B\beta, i) \in D_j \\ (B \to \gamma) \in P \end{cases} \Longrightarrow (B \to \cdot\gamma, j) \in D_j$$

$$\text{Complete: } \begin{cases} (B \to \gamma \cdot, k) \in D_j \\ (A \to \alpha \cdot B\beta, i) \in D_k \end{cases} \Longrightarrow (A \to \alpha B \cdot \beta, i) \in D_j$$

Инициализация: $(S' \to \cdot S, 0) \in D_0$. Слово выводимо в грамматике G тогда и только тогда, когда ситуация $(S' \to S\cdot, 0) \in D_{|w|}$.

Remind.
Ситуация - объект вида $(A \to \alpha \cdot \beta, i)$.
$(A \to \alpha \cdot \beta, i) \in D_j \iff S \vdash_l w_{0i}A\gamma, A \to \alpha\beta \in P, \alpha \vdash_l w_{ij}$

Note: $\vdash_l$ - выводится левосторонним образом

Note: $w_{ij} = w[i, j)$

Remind.
$w \in L(G) \iff (S' \to S\cdot, 0) \in D_{|w|}$

Remind.

Predict: $\forall B \to \gamma \in P \quad \dfrac{(A \to \alpha \cdot B\beta, i) \in D_j}{(B \to \cdot\gamma, j) \in D_j}$

Scan: $\forall a \in \Sigma :\ w[j] = a \quad \dfrac{(A \to \alpha \cdot a\beta, i) \in D_j}{(A \to \alpha a \cdot \beta, i) \in D_{j+1}}$

Complete: $\dfrac{(A \to \alpha \cdot B\gamma, i) \in D_k \qquad (B \to \beta\cdot, k) \in D_j}{(A \to \alpha B \cdot \gamma, i) \in D_j}$

Алгоритм Эрли

```
def earley(word, grammar):
    D_0 = {S' → ·S, 0}
    while D_0 changes:
        D_0 = Complete(D_0)
        D_0 = Predict(D_0)

    for i in range(1, |w|):
        D_i = Scan(D_{i-1}, w_i)
        while D_i changes:
            D_i = Complete(D_i)
            D_i = Predict(D_i)

    return {S' → S·, 0} in D_|w|
```

22. Алгоритм Эрли синтаксического разбора для КС-грамматик: доказательство корректности.

Нужно доказать, что алгоритм правильно строит множества $D_0, \ldots, D_{|w|}$. Это равносильно тому, что он поддерживает инвариант вида $(A \to \alpha \cdot \beta, i) \in D_j \Leftrightarrow \exists \delta \in (\Sigma \cup N)^*((S \vdash w_{0i}A\delta) \wedge \alpha \vdash w_{ij})$.

▲ $\Rightarrow$. Индукция по построению множеств D_j. $(A \to \alpha \cdot \beta, i)$ попадает в D_j в результате некоторой операции: база - $(S \to S', 0)$ удовлетворяет инварианту;

Если операция **Scan**, то $\alpha = \alpha' a$, $a = w[j-1]$ и $(A \to \alpha' \cdot a\beta, i) \in D_{j-1}$. По индукции $S \vdash w_{0i}A\delta$, а значит $\alpha' \vdash w_{i(j-1)}$. Тогда из равенства $a = w[j-1]$ следует $\alpha = \alpha' a \vdash w_{i(j-1)}w[j-1] = w_{ij}$. Чтд.

Если операция **Predict**, то $\alpha = \varepsilon$, $i = j$ - отсюда второй пункт утверждения. Также $\exists i' \leqslant i$ и ситуация $(A' \to \alpha' A \delta', i') \in D_i$, откуда по индукции $S \vdash w_{0i'}A'\delta'', \alpha' \vdash w_{i'i}$. Получаем $S \vdash w_{0i'}A'\delta'' \vdash w_{0i'}\alpha' A\delta'\delta'' \vdash w_{0i'}w_{i'i}A\delta'\delta'' = w_{0i}A\delta$. Чтд.

Если операция **Complete**, то $\alpha = \alpha' A'$ и $\exists i', \delta : (A \to \alpha' \cdot A'\beta, i) \in D_{i'}$, $(A' \to \gamma \cdot, i') \in D_j$. Отсюда $\alpha = \alpha' A' \vdash w_{ii'}w_{i'j} = w_{ij}$; $S \vdash w_{0i}A\delta$ по предположению индукции. Чтд.

$\Leftarrow$. Доказательство индукцией по суммарной длине $w_{0i}A\delta$ из S и w_{ij} из α.

Разбираем случаи в зависимости от α. Если $\alpha = \alpha' a$, то $a = w[j-1]$, $\alpha' \vdash w_{i(j-1)}$. По предположению индукции $(A \to \alpha' \cdot \beta, i) \in D_{j-1}$. Тогда по правилу *Scan* получаем $(A \to \alpha' \cdot \beta, i) \in D_j$

Если $\alpha = \alpha' B$, тогда получаем, что существует i', такой что $\alpha' \vdash w_{ii'}$, $B \vdash w_{i'j}$. Тогда имеем $(A \to \alpha' \cdot B\beta) \in D_{i'}$. Кроме того, можно записать $S \vdash w_{0i}A\delta \vdash w_{0i}w_{ii'}B\beta\delta$, а также $B \vdash_1 \gamma \vdash w_{i'j}$. Применяя индукцию по второму параметру, имеем $(B \to \gamma \cdot, i') \in D_j$, откуда по правилу *Complete* получаем $(A \to \alpha' B \cdot \beta) \in D_j$, что и требовалось.

Пусть теперь $\alpha = \varepsilon$, тогда $i = j$. Тогда либо $i = 0$, $A = S_1$, $\delta = \varepsilon$, что доказывает базу индукции, либо вывод можно переписать в виде $S \vdash w_{0i'}A'\delta'' \vdash_1 w_{0i'}w_{i'i}A\delta'\delta'' = w_{0i}A\delta$ для некоторого правила $(A' \to w_{i'i}A\delta') \in P$. Отсюда по предположению индукции следует, что $(A' \to \cdot w_{i'i}A\delta', i') \in D_{i'}$, что после нескольких применений правила *Scan* приводит к $(A' \to w_{i'i} \cdot A\delta', i') \in D_i$, после чего по правилу *Predict* мы получаем $(A \to \cdot\beta, i) \in D_i$, что и требовалось. Корректность доказана.

■

23. Алгоритм Эрли синтаксического разбора для КС-грамматик: доказательство полноты.

Утверждение: Алгоритм Эрли является полным.

▲ Рассмотрим слово w. Если слово w выводимо в грамматике $G = \langle N, \Sigma, P, S \rangle$, то верно, что $S \vdash w$. Пусть $i = 0$, $j = |w|$. Тогда существуют $\varphi, \psi \in (N \cup \Sigma)^*$, что $\varphi \vdash w[0:0] = \varepsilon$, $S \vdash w[0:|w|] = w$, что $S' \vdash \varphi S' \psi \vdash S' \psi \vdash_1 S\psi$. Укажем явно φ и ψ: $\varphi = \varepsilon$, $\psi = \varepsilon$, тогда выполняется, что:

$$S' \vdash_1 S \vdash w$$

По сути, только что расписали подробно $S' \vdash w$. A соответствует нетерминальному символу S' (вспомогательному стартовому нетерминалу), α соответствует нетерминальному символу S, из которого выводится слово w, β соответствует пустому слову ε. По основной лемме об инварианте, доказанной ранее, ситуация $(S' \to S\cdot, 0) \in D_{|w|}$ тогда и только тогда, когда $S' \vdash w$. Так как из основной леммы следует корректность алгоритма Эрли, то все возможные ситуации будут рассмотрены, и если слово w выводимо в грамматике G, то это эквивалентно тому, что будет рассмотрена ситуация $(S' \to S\cdot, 0) \in D_{|w|}$, и будет выведено, что $w \in L(G)$. Если слово w не является выводимым в грамматике G, то ситуация $(S' \to S\cdot, 0) \notin D_{|w|}$, и будет выведено, что $w \notin L(G)$. Значит, алгоритм Эрли является полным. ■

24. Алгоритм Эрли синтаксического разбора для КС-грамматик: обоснование сложности. Примеры случаев, где Эрли ведёт себя квадратично. Как реализовать алгоритм Эрли, чтобы он был эффективным

Эффективное хранение ситуаций и правил

Требуется эффективным образом хранить ситуации типа $\{A \to \alpha \cdot X\beta, i\} \in D_j$. Множества D_j можно хранить в массиве $D[j][X]$, где $D[j][X] = \{(A_1 \to \alpha_1 \cdot X\beta_1, i_1), (A_2 \to \alpha_2 \cdot X\beta_2, i_2), \dots\}$. Возможны три случая, связанные с символом X:

1. $X = a$, $a \in \Sigma$
2. $X = B$, $B \in N$
3. $X = \$$, $\$$ — конец слова

Правила будем хранить в массиве $G[A]$, где в $G[A]$ будут храниться все правила начинающиеся с A, то есть правила вида $A \to \beta$.

Об операциях

Пусть w — слово.

1. Scan. $\forall j < |w| : w[j] = a$. Тогда нужно рассмотреть все элементы $D[j][a]$ и расположить их в $D[j+1]$ в соответствии с символом, следующим за a. Асимптотика этой операции соответствует $\mathcal{O}(D[j][a])$, то есть она растёт в соответствии с количеством элементов, находящихся в $D[j][a]$.
2. Predict. $\forall j, B$: Пусть ситуация имеет вид $(A \to \alpha \cdot B\beta, i) \in D_j$. Она лежит в $D[j][B]$. Нужно рассмотреть все правила вида $B \to \gamma \in P$, они лежат в $G[B]$. Асимптотика этой операции $O(D[j][B] \cdot G[B])$.
3. Complete. $\forall j, B$: Пусть ситуация имеет вид $(B \to \gamma \cdot, i) \in D_j$, она лежит в $D[j][\$]$. Нужно рассмотреть все ситуации вида $(A \to \alpha \cdot B\beta, k) \in D_i$, они лежат в $D[i][B]$. Асимптотика этой операции $O(\sum_{i \le j} G[B] \cdot D[i][B])$.

Оценки

Оценим, как растёт величина $|D_j|$. $D_j = \{(A_1 \to \alpha_1.\beta_1, i_1), (A_2 \to \alpha_2.\beta_2, i_2), \dots\}$. Пусть $|G|$ — сумма всех длин правых частей правил. Значение i может быть от 0 до j, а точка может быть расположена в $\mathcal{O}(|G|)$ мест, поэтому $\mathcal{O}(|D_j|) = (j+1)|G|$.

Асимптотика операции Scan соответствует $\mathcal{O}(|D_j|) = \mathcal{O}(|w| \cdot |G|)$.
Асимптотика всех операций Scan $\mathcal{O}(|D_j| \cdot |w|) = \mathcal{O}(|w|^2 \cdot |G|)$.

Для оценки асимптотики операции Predict рассмотрим множества вида:

$$(A \to \alpha \cdot B\beta, i) \in D_j$$
$$(B \to \cdot\gamma, j) \in D_j$$

Так как операция Predict сводится к перебору по i, точке и правилам левая часть которых B, то асимптотика операции соответствует $\mathcal{O}(D[j][B] \cdot G[B]) = \mathcal{O}(|w| \cdot |G| \cdot G[B])$.

Асимптотика всех операций Predict $O(\sum_{j,B} D[j][B] \cdot G[B]) = O(|P| \cdot |G| \cdot |w|^2)$, где P - множество всех правил.

Рассмотрим теперь операцию Complete. Ситуации из $D[j][\$]$ указывают на номер множества i и нетерминальный символ B, который соответствует левой части правила из ситуации. Поэтому для каждой ситуации из $D[j][\$]$ будут рассмотрены все ситуации из $D[i][B]$. Поэтому асимптотика всех операций Complete равна $O(\sum_{j,i \le j,B} G[B] \cdot D[i][B]) = O(\sum_{j,i \le j,B} G[B] \cdot D_i) = O(\sum_{j,i \le j} |P| \cdot D_i) = \mathcal{O}(|P| \cdot |w|^2 \cdot |w| \cdot |G|) = \mathcal{O}(|w|^3 |G|^2)$.

Так как итераций $\mathcal{O}(|w|)$, то итоговая сложность алгоритма составляет $\mathcal{O}(|w|^3 |G|^2)$. Однако данную оценку можно сильно улучшить, если грамматически удастся доказать (на лекции не стали доказывать), что количество появлений каждого правила ограничена сверху некоторой константой C, то асимптотика уже будет равна $\mathcal{O}(|w|^2 \cdot C)$. Считаем $|P|, |G|, C$, константами, потому что "вся эта история происходит на стадии компиляции". (в алгоритме мы работаем с одним набором правил, видимо поэтому эти величины можно считать константой)

Есть теорема, что если грамматика однозначна, то алгоритм Эрли для неё стоит квадрат. Идея доказательства в том, что если грамматика однозначна, то в complete в $(A \to \alpha \cdot B\beta, k) \in D_i, (B \to \cdot\gamma, i) \in D_j$ не нужно будет перебирать i. Оно будет подбираться однозначно.

Алгоритм за куб.

Инициализация:
$D_0 \leftarrow \{(S \to \cdot S_1, 0)\}$
for $i = 1, \ldots, |w|$ **do**
  $D_i \leftarrow \varnothing$;
end
Шаг работы:
for $j = 0, \ldots, |w|$ **do**
  $Scan(D, j)$;
  while D_j *изменяется* **do**
    $Complete(D, j)$;
    $Predict(D, j)$;
  end
end
Function *Scan(D, j)*
  if $j = 0$ **then**
    return
  end
  for $(A \to \alpha \cdot a\beta, i) \in D_{j-1}$ **do**
    if $a = w[(j-1)]$ **then**
      D_j.add $((A \to \alpha a \cdot \beta, i))$
    end
  end
end

```
Function Predict(D,j)
    for (A → α · Bβ, i) ∈ D_j do
        for (B → γ) ∈ P do
            D_j.add (B → ·γ, j)
        end
    end
end
Function Complete(D,j)
    for (B → γ·, i) ∈ D_j do
        for (A → α · Bβ, k) ∈ D_i do
            D_j.add (A → αB · β, k)
        end
    end
end
```

Эффективный алгоритм и примеры случаев, где Эрли ведёт себя квадратично: (из neerc.ifmo)

```
function earleyMod(G, w):
    // Инициализация
    D_0 = {[S' → ·S, 0]}
    rulesLoop(0)
    for j = 1 .. n
        for [A → α · a_j β, i] ∈ D_{j-1}
            D_j ∪ = [A → α a_j · β, i]   // Первое правило
        rulesLoop(j)
```

```
function rulesLoop(j):
    D''_j = D_j
    while D''_j ≠ ∅
        D'_j = D''_j
        D''_j = ∅
        for [B → η·, i] ∈ D'_j                 // Цикл (*)
            for [A → α · Bβ, k] ∈ D_i
                D''_j ∪ = [A → αB · β, k]     // Второе правило

        for [B → α · Aη, k] ∈ D'_j             // Цикл (**)
            for β : (A → β) ∈ P
                D''_j ∪ = [A → ·β, j]          // Третье правило
        D_j ∪ = D''_j
```

Если входная грамматика однозначна, то время выполнения алгоритма Эрли для слова длины n составляет $O(n^2)$.

Доказательство:

▷

Орагнизуем каждый список разбора D_j таким образом, чтобы по любому символу $x \in \Sigma \cup N$, можно было за $O(1)$ получить список тех и только тех ситуаций, содержащихся в D_j, которые имеют вид $[A \to \alpha \cdot x\beta, j]$.

Время построения D_0 не зависит от входной строки.

Рассмотрим $D_j,\ j > 0$.

1. При включении ситуаций по правилу (1) необходимо лишь просмотреть предыдущий список и для каждого его элемента выполнить константное число операций.
2. Рассмотрим правило (2). Можно считать, что внутри цикла $(*)$ рассматриваются те и только те ситуации, которые удовлетворяют условию (так как список таких ситуаций можно по нетерминалу получить за $O(1)$ следующим образом: каждый раз, когда мы добавляем ситацаию вида $[A \to \alpha \cdot B\beta, i]$ в D_j, мы просмотрим в заранее заготовленном массиве для D_j, есть ли в D_j ситуации вида $[B \to \eta\cdot, j]$. Если да, то добавим $[A \to \alpha B \cdot \beta, i]$ в D_j.). Тогда каждая такая ситуация будет добавлена в список и, исходя из леммы 2, попытка добавления будет единственной. А так как по лемме 1 всего в списке D_j находится $O(j)$ ситуаций, то суммарно за все итерации внешнего цикла while внутри цикла $(*)$ будет рассмотрено $O(j)$ ситуаций.
3. Так как грамматика фиксирована, то при применении правила (3) при рассмотрении любой ситуации количество включаемых ситуаций не превосходит некоторой константы, поэтому для каждой рассмотренной ситуации будет выполнено $O(1)$ операций.

Таким образом, на построение списка D_j будет потрачено $O(j)$ операций. Тогда время работы алгоритма составляет $O(n^2)$.

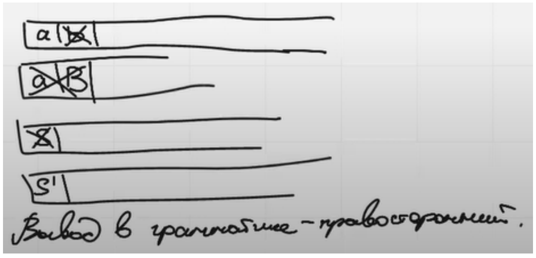
25. Анализатор перенос-свёртка, недостатки анализатора.

Мотивировка: У нас есть алгоритм Эрли, но он работает долго.

- Scan: $O(|w|^2|G|)$
- Predict: $O(|w|^2|G|^2)$
- Complete:$O(|w|^3|G|^2)$

Чтобы понять алгоритм, **рассмотрим грамматику и слово w = ab**

- $S' \to S$(добавляем)
- $S \to aB$
- $B \to b$



Храним стек текущих символов и текущую позицию в слове.

- На вершине стека написана правая часть правила - заменяем на левую (reduce)
- Иначе - добавляем символ в стек и читаем символ (shift)

Стоит заметить, что разбираем мы слово справа налево, а вывод в грамматике правсторонний, поэтому мы будем вынуждены поменять порядок правил на обратный для построения дерева разбора

А в чем проблема?

1 А как понять, какая правая часть правила находится на стеке?

2 А что делать, если нам подходят несколько правил? (стек можно распарсить двумя и более способами)

26. Определение LR-грамматики, примеры LR и не LR-грамматик. Стековая аналогия определения (нет конфликтов). Вопрос на отл: однозначность lr грамматики.

Опеределение: Грамматика называется LR(k) грамматикой, если LR-k таблица для этой грамматики строится корректно

Опеределение: Для $\alpha \in (N \cup \Sigma)^*$ определим First(α) как:

First(α) $= \{a | \alpha \ \to au, u \in \Sigma^*\} \cup \{\$\} I(\alpha \to \varepsilon)$ First(α) - первый символ, который может вывестись из α.

Если $\alpha = a\gamma$, то First(α) = a

Если $\alpha =$ Bγ, B $\to \varepsilon$, то First(α) = First(B) $\cup$ First(γ)

Опеределение: Грамматика называется LR(k) грамматикой, если из условий:

1. $S' \to \cdot\alpha A w \to \alpha\beta w$
2. $S' \to \cdot\gamma B x \to \alpha\beta y$ где γ - префикс α, возможно, несобственный. x - суффикс w
3. $First_k(w) = First_k(y)$

Что происходит в теминах алгоритма? На стеке сейчас лежат $\alpha\beta$ и мы смотрим, верно ли, что можно свернуть однозначно? Наличие двух одинаковых выводов говорит, что нет. То есть алгоритм стоит перед некоторой дилеммой. Можно свернуть A (по правилу $A \to \beta$) или прочитать что-то еще, и свернуть B(по правилу $B \to \nu_1\beta\nu_2$). В первлм случае мы сворачиваем за один шаг, во втором случае - что-то еще сворачиваем, а уже после сворачиваем B.

Что делать в таких ситуациях? А ничего. мы говорим, что такого не допускаем :)

То есть мы говорим, что $\alpha A y = \gamma B x$, то есть $\alpha = \gamma, A = B$ **Конец определения**

Примеры LR грамматик

- Грамматика из примера к предыдущему билету
-
  - $S \to A$
  - $A \to aA \mid b$

  - $S \to A \mid aB$
  - $A \to aA \mid r$
  - $B \to aB \mid m$

Эти грамматики парсятся LR(0), и, соответственно, любой другой грамматикой

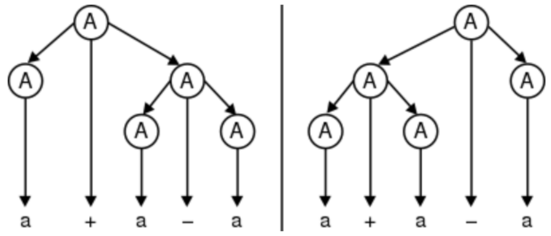
Любая неоднозначная грамматика - не LR грамматика

A → A + A | A - A | a

неоднозначно, поскольку есть два крайних левых вывода для строки a + a + a:

A → A + A	A → A + A
→ a + A	→ A + A + A (Первый A заменяется на A + A. Замена второго A приведет к аналогичному выводу)
→ a + A + A	→ a + A + A
→ a + a + A	→ a + a + A
→ a + a + a	→ a + a + a

В качестве другого примера грамматика неоднозначна, поскольку существует два дерева синтаксического анализа для строка a + a - a:



Язык, который она генерирует, однако, не является неоднозначным по своей сути; следующая однозначная грамматика порождает тот же язык:

A → A + a | A - a | a

Дополнительно. Пример грамматики LR(1), не являющейся LR(0) грамматикой

- $S' \to S$
- $S \to Bb$
- $S \to Cc$
- $B \to a$
- $C \to a$

LR(0) разбор

0
- $S' \to \cdot S$
- $S \to \cdot Bb$
- $S \to \cdot Cc$
- $B \to \cdot a$
- $C \to \cdot a$

1
- $B \to a \cdot$
- $C \to a \cdot$

И все, есть два reduce. Мы не знаем, что делать. А в терминах вывода это означает, что мы могли открыть букву как ab и как ac, и неважно, что слова разные, у нас уже конфликт в явном виде

LR(1) разбор

0
- $S' \to \cdot S, \$$
- $S \to \cdot Bb, \$$

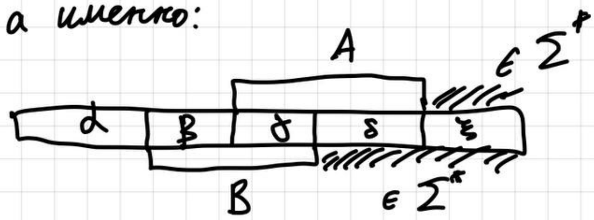
- $S \to \cdot Cc\$$
- $B \to \cdot a, b$
- $C \to \cdot a, c$

1
- $B \to a\cdot, b$
- $C \to a\cdot, c$

  Тут такого противоречия нет, так как мы подсмотрели следующие буквы и поняли, что это соответственно будут reduce(3) и reduce(4), то есть сейчас мы эти ситуации отличаем

Однозначность LR-грамматики (на отл):

$\boxed{2}$ Пусть G неоднозначна, т.е. есть слово w с двумя различными правостор. выводами. Рассм. последний момент, когда концы выводов совпадают, т.е. в обратном правост. выводе мы по-разному свернули подстроку текущей цепочки, а именно:



$S' \vdash \alpha\beta A\xi \to \alpha\beta\gamma\delta\xi$

$S' \vdash \alpha B\delta\xi \to \alpha\beta\gamma\delta\xi$

Обращаем внимание, что мы получаем опр-ие $LR(k)$-грам-ки (подстроки выделены для наглядности). Но по определению $A = B$, $\xi = \delta\xi$, $\alpha\beta = \alpha$ $\Rightarrow$ в обоих выводах использовалось правило $A \to \gamma$ для строки $\alpha A \xi$, хотя по предположению выводы в этом моменте отличаются $\Rightarrow$ грам-ка однозначна

27. Алгоритм построения LR-таблицы. Вопрос на отл: доказательство корректности и полноты. Если верно определение, таблица строится и если не верно, не строится. Критерий LR-овости (отсутствие противоречий).

Определение: $A \to \alpha \cdot \beta, a_1 \dots a_k$ - LR(k)-ситуация (неформально: если находимся в $A \to \alpha \cdot \beta$, то $a_1 \dots a_k$ - первые буквы того, что можем вывести дальше, $a_i \in \Sigma \cup \{\$\}$)

Замечание: Далее будем рассматривать алгоритм LR(1), для $k \neq 1$ действуем аналогично.

Определение: Пусть I - множество ситуаций. Тогда $CLOSURE(I) := J, I \subset J$, такое что

$$\left.\begin{array}{c} B \to \gamma \in P \\ A \to \alpha_1 \cdot B\alpha_2, a \in J \end{array}\right\} \Rightarrow \forall c \in First(\alpha_2 a): \; B \to \cdot\gamma, c \in J$$

Определение: Пусть I - множество ситуаций, $\lambda \in \Sigma \cup N$. Тогда

$$GOTO(I, \lambda) := CLOSURE(\{\langle A \to \alpha_1\lambda \cdot \alpha_2\rangle \mid \langle A \to \alpha_1 \cdot \lambda\alpha_2\rangle \in I\})$$

```
item[] closure(item[] I):
    bool changed
    item[] J = I
    repeat
        changed = false
        for [A → α · Bβ, a] ∈ I
            for (B → γ) ∈ Γ'.P
                for b ∈ FIRST(βa)
                    J.add([B → ·γ, b])
                    changed = true
    until not changed
    return J
```

```
item[] goto(item[] I, char X):
    item[] J = ∅
    for [A → α · Xβ, a] ∈ I
        J.add([A → αX · β, a])
    return closure(J)
```

Построим ДКА, вершинами которого будут множества ситуаций, а на ребрах будут написаны символы из $\Sigma \cup N$, по которым будем делать $GOTO$. Как строим: в начальное состояние добавляем $S' \to \cdot S, \$$. Затем берем $CLOSURE$ от состояния, делаем $GOTO$ по всем возможным буквам - получаем новые вершины автомата, в которые ведут ребра по этим буквам. Повторяем процесс для добавленных состояний и тд.

```
item[][] items(Γ'):
    bool changed
    item[][] C
    C.add(closure({[S' → ·S, $]}))
    repeat
        changed = false
        for item[] I ⊂ C
            for X ∈ Γ'.Σ
                if goto(I, X) ≠ ∅ and goto(I, X) ⊄ C
                    C.add(goto(I, X))
                    changed = true
    until not changed
    return C
```

Теперь нам остается только построить таблицу *table* по автомату (столбцы - символы из $N \cup \Sigma \cup \{\$\}$, строки - номера состояний в автомате):

1. Если из состояния i в состояние j ведет ребро по букве a, то $table[i][a] = (shift, j)$.
2. Если в состоянии i есть ситуация $A \to \alpha\cdot, a$, то $table[i][a] = (reduce, j)$, где j - номер правила $A \to \alpha$ в грамматике. Если нужно записать $(reduce, 0)$, то есть свертка по правилу $S' \to S$, то записываем $table[i][a] = accept$
3. Если никакой из прошлых пунктов не записался, то $table[i][a] = error$

```
// вход: Γ' – расширенная грамматика
// выход: таблица T канонического LR(1)-анализа
function getLR1CanonicalTable(Γ') :
   C'(Γ') ← {I_0, I_1..I_n} // множество канонических ситуаций для Γ'
   fillArray(T, Error):
   foreach
       if [A → α · aβ, b] ∈ I_i and goto(I_i, a) = I_j // здесь a – терминал
           T[i, a] = Shift(j)
       if [A → α·, a] ∈ I_i and A ≠ S'
           T[i, a] = Reduce(A → a)
       if [S' → S·, $] ∈ I_i
           T[i, $] = Accept
       if goto(I_i, A) = I_j
           goto[i, A] ← j
```

28. Алгоритм разбора по LR-таблице. Структура стека. Корректность и полнота разбора (на отл).

Пусть у нас есть готовая LR-таблица table и дано слово $w = w_1 \dots w_n\$$. Создаем стек: изначально записываем в него 0 - номер правила $S' \to S$, так как оно всегда первое в нашем выводе. Дальше запускаем цикл

1. Если последний символ на стеке - это номер правила (i), то пытаемся дописать на стек следующую букву слова w_j: проверяем, если в ячейке $[i, w_j]$ стоит пометка $shift$ (то есть из текущего состояния есть переход по этой букве), то дописываем эту букву на стек, иначе говорим, что слово не лежит в языке.

2. Берем последние 2 символа со стека: среди них гарантированно одна буква (c) и одно число (k). Смотрим в ячейку $[k, c]$

   (a) Если там стоит $(shift, to)$, то дописываем на стек to - номер состояния в который надо перейти

   (b) Если там стоит $(reduce, rule)$, то находим правило под номером $rule$, удаляем со стека $2 \cdot |rule.right|$ символов (нужно удалить все буквы из этого правила, но на стеке они чередуются с цифрами, поэтому умножаем длину на 2) и дописать $rule.left$. Если делаем свертку по 0 правилу $S' \to S$, то есть в этой ячейке стоит Accept, то говорим, что слово лежит в языке.

   (c) Если там стоит Error, то слово не лежит в языке

Из алгоритма видно, что на стеке чередуются буквы и числа (в самом начале число)

4. Конечные преобразователи

29. Конечные преобразователи и задаваемые ими преобразования. Различные варианты определения. Примеры конечных преобразований. Теорема Нива.

Определение: Конечный преобразователь $T = \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle$:

1. Q — множество состояний, $|Q| < \infty$;
2. Σ — входной алфавит, $|\Sigma| < \infty$;
3. Γ — выходной алфавит, $|\Gamma| < \infty$;
4. $\Delta \subset (Q \times \Sigma^*) \times (Q \times \Gamma^*)$ — множество переходов;
5. $q_0 \in Q$ — стартовое состояние;
6. $F \subset Q$ — множество завершающих состояний. Можно считать, что $|F| = 1$

Определение: Конфигурация КПтеля T – это тройка $\langle q, u, v \rangle \quad (q \in Q, u \in \Sigma^*, v \in \Gamma^*)$
Неформально: находимся в состоянии q; осталось прочесть слово u; вывели слово v

Определение: Отношение выводимости ($\vdash$) – наименьшее рефлексивное транзитивное отношение, что:

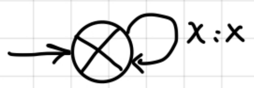
$$\forall \langle q_1, u \rangle \to \langle q_2, v \rangle \in \Delta \text{ выполнено: } \forall y \in \Sigma^*,\ z \in \Gamma^* : \langle q_1, uy, z \rangle \vdash \langle q_2, y, zv \rangle$$

Определение: Соответсвие, задаваемое конечным преобразователем T:

$$\psi = \{(u, v) \in \ | \ \exists q \in F \ : \ \langle q_0, u, \varepsilon \rangle \vdash \langle q, \varepsilon, v \rangle\}$$

Определение: Конечное преобразование (КП) – это $\psi : \Sigma^* \to \Gamma^*$

Примеры:

1) id $(\Gamma = \Sigma)$ = echo 

$x \in \Sigma$

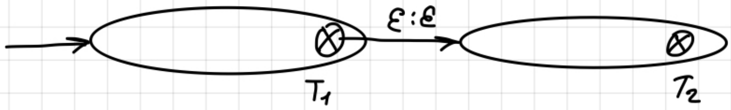
2) $\psi : \{(u, \varepsilon) : u \in R\}$ R- регулярный язык

$\langle q_1, a \rangle \xrightarrow[M]{} q_2 \Longrightarrow \langle q_1, a \rangle \xrightarrow[T]{} \langle q_2, \varepsilon \rangle$

Аналогично, $\psi = \{(\varepsilon, v) : v \in R\}$

3) Конкатенация $\psi_i = \{(u_i, v_i)\}$ $i = 1,2$

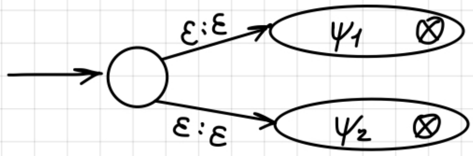
Тогда $\psi = \psi_1 \psi_2 = \{(u_1 u_2, v_1 v_2)\}$



4) $\psi = w \to \{wu : u \in R\}$ $= wR^*$

Это $\psi = id \circ \{(\varepsilon, u) : u \in R\}$

5) ψ_1 - КП, ψ_2 - КП $\Rightarrow$ $\psi_1 \cup \psi_2$ - КП



6) Пусть φ - гомоморфизм : $\Sigma \to \Gamma$

т.е. $\varphi(uv) = \varphi(u)\varphi(v)$

Пример: солненый язык

каша → касашаса т.к φ(согл) = согл; φ(гл) = гл·с·гл

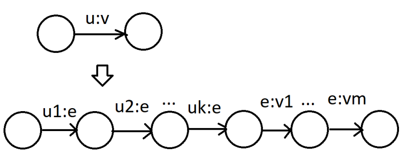
Опеределение: Неудлиняющий гомоморфизм – это $\psi : \forall x \in \Sigma^* \;\; |\psi(x)| \leqslant |x|$

Лемма: $\forall T =< Q, \Sigma, \Gamma, \Delta, q_0, F > \exists$ эквивалентный ему $T' =< Q', \Sigma, \Gamma, \Delta', q_0, F' >$:

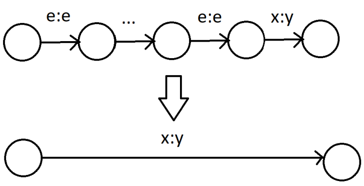
$\forall < q_1, u > \to < q_2, v > \in \Delta' : |u| + |v| = 1$.

▲ Аналогично НКА.

$u = u_1 u_2 \dots u_k, v = v_1 v_2 \dots v_m$. Добавим состояния и переходы вида $u_i : \varepsilon$ и $\varepsilon : v_j$.



Осталось убрать переходы вида $\varepsilon : \varepsilon$ (не забываем корректно проставить завершающие состояния) :



■

Теорема Нива: Любое конечное преобразование можно представить в виде композиции:

- обратного неудлиняющего гомоморфизма: $\psi^{-1} : \Theta^* \to \Sigma^*$
- ограничения на регулярный язык: $id_R \;\; R \subset \Theta^*$
- неудлиняющего гомоморфизма: $\eta : \Theta^* \to \Gamma^*$

▲ Если ψ – КП, то существует конечный преобразователь $M : L(M) = \psi$ т.ч. $|u| + |v| = 1$.

Пронумеруем ребра конечного преобразователя $M : e_1, e_2, \dots, e_n$

$\Theta = \{e_1, e_2, \dots, e_n\}$

R – все пути из q_0 в F. Заметим, что язык автоматный т.к. на каждом ребре написан символ, ему соответствующий. Поэтому он и регулярный.

ϕ – для каждого ребра берем вход, η – для каждого ребра берем выход

Заметим, что ϕ и η неудлиняющие т.к. $|\phi(x)| \leq 1 \forall x \in \Theta$ т.к. на переходах не более одной буквы.

Докажем теперь про композицию.

$\psi = (u, v)$

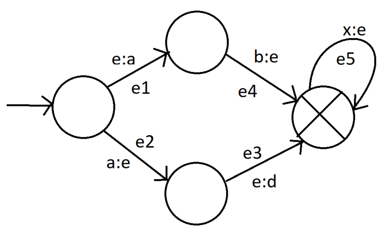
Рассмотрим автомат над Θ, который мы получаем, когда расписываем все по однобуквенным переходам и нумеруем ребра. В нем есть какой-то путь из q_0 в $q \in F$, пройдя по которому, мы u преобразуем в v, то есть:

$\langle q_0, e_{i1} \dots e_{ik} \rangle \vdash \langle q, \varepsilon \rangle$. Заметим, что тогда $e_{i1}, \dots, e_{ik} \in R$, поэтому id его не изменит (оставит).

$\phi(e_{i1} \dots e_{ik}) = u, \;\; \eta(e_{i1} \dots e_{ik}) = v$ (из определения ϕ и η).

Поэтому для $\psi = (u, v) \Rightarrow \exists q \in F : \langle q_0, u, \varepsilon \rangle \vdash_k \langle q, \varepsilon, v \rangle$. ■

Пример:



$\Theta = e_1, e_2, e_3, e_4, e_5,$

$R = (e_1 e_4 + e_2 e_3) e_5^*$

$\phi(e_1) = \varepsilon, \phi(e_2) = a, \phi(e_3) = \varepsilon, \phi(e_4) = b, \phi(e_5) = x$

$\eta(e_1) = a, \eta(e_2) = \varepsilon, \eta(e_3) = d, \eta(e_4) = \varepsilon, \eta(e_5) = \varepsilon$

30. Замкнутость конечных преобразований относительно композиции.

Теорема: Если $\psi_1 : \Sigma^* \to \Gamma^*$ и $\psi_2 : \Gamma^* \to \Pi^*$ – КП, то $\psi = \psi_2 \circ \psi_1$ – КП

▲ $\psi_1 = L(M_1)$, M_1 – КПтель : $\langle Q, \Sigma, \Gamma, \Delta_Q, q_0, F_Q \rangle$

$\psi_1 = L(M_1)$, M_1 – КПтель : $\langle P, \Gamma, \Pi, \Delta_P, p_0, F_P \rangle$

Тогда $M_{comp} = \langle Q \times P, \Sigma, \Pi, \Delta', (q_0, p_0), (F_Q \times F_P) \rangle$

Сразу отметим, что все переходы в ψ_1, ψ_2 : $|u| + |v| = 1$. Тогда положим:

читаем с stdin

$$\Delta' = \{ \langle (q_1, p), a \rangle \to \langle (q_2, p), \varepsilon \rangle \mid \langle q_1, a \rangle \to \langle q_2, \varepsilon \rangle \in \Delta_Q \}$$

$$\cup \{ \langle (q_1, p_1), \varepsilon \rangle \to \langle (q_2, p_2), \varepsilon \rangle \mid \langle q_1, \varepsilon \rangle \to \langle q_2, \alpha \rangle \in \Delta_Q, \ \langle p_1, \alpha \rangle \to \langle p_2, \varepsilon \rangle \in \Delta_P \}$$

перенаправление инфы

$$\cup \{ \langle (q, p_1), \varepsilon \rangle \to \langle (q, p_2), a \rangle \mid \langle p_1, \varepsilon \rangle \to \langle p_2, a \rangle \in \Delta_P \}$$

добавляем в stdout

Лемма: $\langle (q_1, p_1), u, \varepsilon \rangle \vdash_{M_{comp}} \langle (q_2, p_2), \varepsilon, v \rangle \iff \exists w \in \Gamma^* : \langle q_1, u, \varepsilon \rangle \vdash_{M_Q} \langle q_2, \varepsilon, w \rangle$ и $\langle p_1, w, \varepsilon \rangle \vdash_{M_P} \langle p_2, \varepsilon, v \rangle$

$\Longrightarrow$: индукция по длине вывода в M_{comp}. В переходе делаем разбор случаев: (a, ε), $(\varepsilon, \varepsilon)$, (ε, a)

▲ *База:* 0 шагов

$\langle (q_1, p_1), u, \varepsilon \rangle \vdash_0 \langle (q_2, p_2), \varepsilon, v \rangle$.

Значит, $q_1 = q_2, p_1 = p_2, u = \varepsilon, v = \varepsilon$

Переход:

Мы имеем следующее: $\langle (q_1, p_1), u, \varepsilon \rangle \vdash \langle (q_3, p_3), u', v' \rangle \vdash_1 \langle (q_2, p_2), \varepsilon, v \rangle$

Далее у нас есть 3 случая в зависимости от того, какой был последний переход (они соответствуют трем множествам из объединения для Δ):

1. Можем сделать несколько выводов:
   (a) $u' = a$
   (b) $v' = v$
   (c) $p_3 = p_2$
   (d) $\langle q_3, a \rangle \to \langle q_2, \varepsilon \rangle \in \Delta_p$ (исходное Δ)

   Поэтому $\langle (q_1, p_1), u''a, \varepsilon \rangle \vdash \langle (q_3, p_2), a, v \rangle \Rightarrow \langle (q_1, p_1), u'', \varepsilon \rangle \vdash \langle (q_3, p_2), \varepsilon, v \rangle$.

   Значит, по предположению индукции, $\exists w : \langle q_1, u'', \varepsilon \rangle \vdash \langle q_3, \varepsilon, w \rangle$ и $\langle p_1, w, \varepsilon \rangle \vdash \langle p_2, \varepsilon, v \rangle$. (Получили одну из нужных выводимостей).

   А еще $\langle q_1, u''a, \varepsilon \rangle \vdash \langle q_3, a, w \rangle \vdash \langle q_2, \varepsilon, w \rangle$.

2. Можем сделать несколько выводов:

   (a) $u' = \varepsilon$

   (b) $v' = v$

   (c) $\exists \alpha \in \Gamma :$
   $\langle q_3, \varepsilon \rangle \to \langle q_2, \alpha \rangle$ и $\langle p_3, \alpha \rangle \to \langle p_2, \varepsilon \rangle$

   Из $\langle (q_1, p_1), u, \varepsilon \rangle \vdash \langle (q_3, p_3), \varepsilon, v \rangle$ по предположению индукции следует, что

   $\exists w \in \Gamma^* :$

   $\langle q_1, u, \varepsilon \rangle \vdash \langle q_3, \varepsilon, w \rangle$ – (1)

   $\langle p_1, w, \varepsilon \rangle \vdash \langle p_3, \varepsilon, v \rangle$ – (2)

   Применим к (1) результаты вывода номер 3 и получим: $\langle q_3, \varepsilon, w \rangle \vdash \langle q_2, \varepsilon, w\alpha \rangle$

   А, воспользовавшись уже второй выводимостью из вывода номер 3 получим:

   $\langle p_1, w\alpha, \varepsilon \rangle \vdash \langle p_3, \alpha, v \rangle \vdash \langle p_2, \varepsilon, v \rangle$

   Из существования $w\alpha$ – переход выполнен.

3. Можем сделать несколько выводов:

   (a) $q_3 = q_2$

   (b) $v = v'a$

   (c) $u' = \varepsilon$

   (d) $\langle p_3, \varepsilon \rangle \to \langle p_2, a \rangle \in \Delta_p$

   Из $\langle (q_1, p_1), u, \varepsilon \rangle \vdash \langle (q_2, p_3), \varepsilon, v' \rangle$ по предположению индукции следует, что

   $\exists w \in \Gamma^* :$

   $\langle q_1, u, \varepsilon \rangle \vdash \langle q_2, \varepsilon, w \rangle$ – (1)

   $\langle p_1, w, \varepsilon \rangle \vdash \langle p_3, \varepsilon, v' \rangle$ – (2)

   Из (2) и вывода номер 4: $\langle p_3, \varepsilon, v' \rangle \vdash \langle p_2, \varepsilon, v'a \rangle$ ($v'a = v$, поэтому переход готов)

■

$\Longleftarrow$: индукция по сумме длин выводов в M_Q и M_P. В переходе расписываем чтение буквы a и написание α

▲ *База:* $k = 0, m = 0$

Тогда $q_1 = q_2, u = \varepsilon, w = \varepsilon, v = \varepsilon, p_1 = p_2$. Тогда утверждение, очевидно, верно.

Переход: И снова случаи

1. Чтение буквы a.

   $\exists w : \langle q_1, u, \varepsilon \rangle \vdash \langle q_3, a, w \rangle \vdash \langle q_2, \varepsilon, w \rangle$ – (1)

   Далее получим 2 выводимости:

$\langle p_1, w, \varepsilon\rangle \vdash \langle p_2, \varepsilon, v\rangle$ и $\langle q_1, u', \varepsilon\rangle \vdash \langle q_3, \varepsilon, w\rangle$.

Применим к ним предположение индукции:

$\langle (q_1, p_1), u', \varepsilon\rangle \vdash \langle (q_3, p_2), \varepsilon, v\rangle$

Тогда (из свойств нашего перехода и (1)):

$\langle (q_1, p_1), u'a, \varepsilon\rangle \vdash \langle (q_3, p_2), a, v\rangle \vdash \langle (q_2, p_2), \varepsilon, v\rangle$.

Переход доказан.

2. Пишем α и читаем α Мы имеем следующее $\exists w = a'\alpha : \langle q_1, u, \varepsilon\rangle \vdash \langle q_3, \varepsilon, w'\rangle \vdash \langle q_2, \varepsilon, w'\alpha\rangle$ – (1)

   $\langle p_1, w'\alpha, \varepsilon\rangle \vdash \langle p_2, \varepsilon, v\rangle$ – (2)

   Из (1) получим $(q_3, \varepsilon) \to (q_2, \alpha)$, а из (2) получим $(p_3, \alpha) \to (p_2, \varepsilon)$.

   Тогда из них можно собрать мощное правило $\langle (q_3, p_3), \varepsilon\rangle \to \langle (q_2, p_2), \varepsilon\rangle$

   Применив предположение индукции и это правило получим

   $\langle (q_1, p_1), u, \varepsilon\rangle \vdash \langle (q_3, p_3), \varepsilon, v\rangle \vdash \langle ((q_2, p_2)), \varepsilon, v\rangle$

3. Пишем букву a. Аналогично случаю 2.

■

Покажем, почему $Lemma \Longrightarrow Theorem$:

$$(u, v) \in \psi \iff \exists (q, p) \,:\, q \in F_Q,\ p \in F_P \text{ т.ч. } \langle (q_0, p_0), u, \varepsilon\rangle \vdash \langle (q, p), \varepsilon, v\rangle \overset{lemma}{\iff}$$

$$\exists w : \langle q_0, u, \varepsilon\rangle \vdash_{M_Q} \langle q, \varepsilon, w\rangle \text{ и } \langle p_0, w, \varepsilon\rangle \vdash_{M_P} \langle p, \varepsilon, v\rangle \iff$$

$$\exists w \,:\, (u, w) \in \psi_1 \text{ и } (w, v) \in \psi_2 \iff (u, v) \in \psi_2 \circ \psi_1$$

■

31. Замкнутость класса автоматных языков относительно конечных преобразований.

Теорема.

Если L – автоматный язык, ψ – конечное преобразование, то $\psi(L)$ – автоматный язык.

Доказательство

Так как ψ – конечное преобразование, то по теореме Нива существуют такие η, R, φ, что $\psi = \eta \circ id|_R \circ \varphi^{-1}$, где η, φ – неудлиняющие гомоморфизмы, R – регулярный (автоматный) язык.

Введём обозначения:

1) $\varphi^{-1}(L) = L_1$

2) $id|_R (L_1) = L_2$

3) $\eta(L_2) = L_3$.

Тогда нужно доказать, что L_1, L_2, L_3 – автоматные языки.

1) φ – неудлиняющий гомоморфизм, а значит $\forall x \in \Sigma \ \varphi(x) = \varepsilon$ или $\varphi(x) = a \in \Sigma$

Обозначим

$\Gamma_\varepsilon = \{u \in \Sigma \mid \varphi(u) = \varepsilon\}$

$\Gamma_a = \{u \in \Sigma \mid \varphi(u) = a\}$

Так как L – автоматный язык, то для него существует задающий его автомат, на рёбрах которого написана ровно одна буква. Преобразуем этот автомат так, чтобы он распознавал L_1:

У каждой вершины сделаем петли по Γ_ε

Если был переход по a, то заменим его переходами по Γ_a.

2) $L_2 = L_1 \cap R$. Воспользуемся тем, что пересечение автоматных языков – автоматный язык.

3) Построим автомат для L_2 и заменим переходы по a на $\eta(a)$.

32. Замкнутость класса контекстно-свободных языков относительно конечных преобразований.

Теорема.

Если L – контекстно-свободный язык, ψ – конечное преобразование, то $\psi(L)$ – контекстно-свободный язык.

Доказательство

Так как ψ – конечное преобразование, то по теореме Нива существуют такие η, R, φ, что $\psi = \eta \circ id|_R \circ \varphi^{-1}$, где η, φ – неудлиняющие гомоморфизмы, R – регулярный (автоматный) язык.

Тогда достаточно доказать 3 утверждения:

1) Если φ – неудлиняющий гомоморфизм и L – КС-язык, то $\varphi^{-1}(L)$ – КС-язык.

2) Если R – регулярный язык и L – КС-язык, то $id|_R(L) = L \cap R$ – КС-язык.

3) Если φ – неудлиняющий гомоморфизм и L – КС-язык, то $\varphi(L)$ – КС-язык.

1) Введем обозначения:

$\Gamma_\varepsilon = \{u \in \Gamma \mid \varphi(u) = \varepsilon\}$

$\Gamma_a = \{u \in \Gamma \mid \varphi(u) = a\}$

Тогда $\varphi^{-1}(\varepsilon) = \Gamma_\varepsilon^*$, а $\varphi^{-1}(a) = \Gamma_\varepsilon^* \Gamma_a \Gamma_\varepsilon^*$

Заведём нетерминалы S_a:

$S_a \vdash \varphi^{-1}(a) = \Gamma_\varepsilon^* \Gamma_a \Gamma_\varepsilon^*$

Так как $\Gamma_\varepsilon^* \Gamma_a \Gamma_\varepsilon^*$ – регулярный, то построим $G_a = \langle \ldots, S_a \rangle$, распознающий этот язык.

L – КС-язык, следовательно, существует КС-грамматика в нормальной форме Хомского $G = \langle N, \Sigma, P, S \rangle \; : \; L(G) = L$, то есть в ней есть только правила вида $S \to \varepsilon, A \to BC, A \to a$.

Заменим правила в ней следующим образом:

$S \to \varepsilon$ заменим на $S \to S_\varepsilon$, при этом добавим правила $S_\varepsilon \to x S_\varepsilon \forall x \in \Gamma_\varepsilon$ и $S_\varepsilon \to \varepsilon$.

$S \to AB$ заменим на $S \to S_\varepsilon A S_\varepsilon B S_\varepsilon$.

$A \to a$ заменим на $A \to S_a$

2) Если L – КС-язык, R – регулярный язык, то $L \cap R$ – КС-язык.

Пусть R – регулярный язык, заданный своим ДКА $\langle \Sigma, Q_1, s_1, T_1, \delta_1 > \rangle$, и L – КС-язык, заданный своим МП-автоматом: $\langle \Sigma, \Gamma, Q_2, s_2, T_2, z_0, \delta_2 \rangle$. Тогда прямым произведением назовем следующий автомат:

- $Q = \{\langle q_1, q_2 \rangle \mid q_1 \in Q_1, q_2 \in Q_2\}$. Иначе говоря, состояние в новом автомате – пара из состояния первого автомата и состояния второго автомата.
- $s = \langle s_1, s_2 \rangle$
- Стековый алфавит Γ остается неизменным.

- $T = \{\langle t_1, t_2 \rangle \mid t_1 \in T_1, t_2 \in T_2\}$. Допускающие состояния нового автомата – пары состояний, где оба состояния были допускающими в своем автомате.

- $\delta(\langle q_1, q_2 \rangle, c, d) = \langle \delta_1(q_1, c), \delta_2(q_2, c, d) \rangle$. При этом на стек кладется то, что положил бы изначальный МП-автомат при совершении перехода из состояния q_2, видя на ленте символ c и символ d на вершине стека.

Этот автомат использует в качестве состояний пары из двух состояний каждого автомата, а за операции со стеком отвечает только МП-автомат. Слово допускается этим автоматом $\Leftrightarrow$ слово допускается и ДКА, и МП-автоматом, то есть язык данного автомата совпадает с $R \cap L$.

3) L – КС-язык, следовательно, существует КС-грамматика в нормальной форме Хомского $G = \langle N, \Sigma, P, S \rangle \ : \ L(G) = L$, то есть в ней есть только правила вида $S \to \varepsilon, A \to BC, A \to a$. Для того, чтобы эта грамматика задавала язык $\varphi(L)$ достаточно заменить правила вида $A \to a$ на $A \to \varphi(a)$ (остальные правила оставить без изменений).

$S \vdash_G w \Leftrightarrow S \vdash_{G'} \varphi(w)$

Идея доказательства: по индукции можно доказать, что

$A \vdash_G w \Leftrightarrow A \vdash_{G'} \varphi(w)$.

33. Лемма о разрастании для конечных преобразований. Примеры соответствий, не задаваемых конечными преобразованиями.

Лемма.

Пусть ψ – конечное преобразование. Тогда:

$$\exists p: \ \forall (u,v) \in \psi : |u| + |v| \geq p \ \exists x_1, y_1, z_1 \in \Sigma^*, x_2, y_2, z_2 \in \Gamma^* \ : \ u = x_1 y_1 z_1, v = x_2 y_2 z_2 :$$

$$|y_1| + |y_2| > 0, |x_1 y_1| + |x_2 y_2| \leq p \ : \forall k \in \mathbb{N} \ (x_1 y_1^k z_1, x_2 y_2^k z_2) \in \psi$$

Доказательство

Пусть M – конечный преобразователь с однобуквенными переходами, задающий конечное преобразование ψ.

Положим $p = |Q|$. Условие $|u| + |v| \geq p$ означается, что мы посетили $\geq p + 1$ состояние, а значит $\exists q \in Q$, которое посетили дважды (если их несколько, рассмотрим самую первую).

Пусть $\langle q_0, x_1 y_1 z_1, \varepsilon \rangle \vdash \langle q, y_1 z_1, x_2 \rangle$, $\langle q, y_1 z_1, x_2 \rangle \vdash \langle q, z_1, x_2 y_2 \rangle$, $\langle q, z_1, x_2 y_2 \rangle \vdash \langle q_f, \varepsilon, x_2 y_2 z_2 \rangle$.

Тогда $|y_1| + |y_2| > 0$ (так как M – конечный преобразователь с однобуквенными переходами) и $|x_1 y_1| + |x_2 y_2| \leq p$ (иначе q – не первое пересечение по состояниям, так как нашёлся бы другой цикл).

Тогда из-за условия $\langle q, y_1 z_1, x_2 \rangle \vdash \langle q, z_1, x_2 y_2 \rangle$ можно бесконечно "накачивать"y_1 и y_2. Более формально, $\forall k \in \mathbb{N}$ $(x_1 y_1^k z_1, x_2 y_2^k z_2) \in \psi$.

Пример соответствия, не задаваемого конечным преобразованием

$\psi : w \to w^R$

Докажем, что это преобразование нельзя задать конечным преобразованием.

Зафиксируем произвольное p. Тогда $(a^p b^p, b^p a^p) \in \psi$.

$a^p b^p = x_1 y_1 z_1, b^p a^p = x_2 y_2 z_2$

$|x_1 y_1| + |x_2 y_2| \leq p \Rightarrow x_1 y_1 = a^l, x_2 y_2 = b^m$

Пусть $y_1 = a^t, t > 0$. Тогда $x_1 y_1^2 z_1 = a^{p+t} b^p$.

$|x_2 y_2^2 z_2|_a = p < p + t$, то есть $(x_1 y_1^2 z_1, x_2 y_2^2 z_2) \notin \psi$.