Билеты по теоринфе$^{\beta}$

3 семестр
@keba4ok

2 января 2022г.

Жёстко записываем все билеты.

Содержание

Охотинская глава.

Да, я знаю, что всё это есть в конспектах Александра Сергеевича, мне просто удобно готовиться, когда всё лежит в одном месте. Не нравится - не читайте и не бурлите.

Билет 1.

Машины Тьюринга. Множество, не распознаваемое никакой машиной Тьюринга. Неразрешимые задачи.

Лекция 1. Введение в предмет ТИ: машины Тьюринга, неразрешимые задачи.

Конспект, запись.

Определение 1. *Машина Тьюринга* - это семёрка $M = (\Sigma, \Gamma, Q, q_0, \delta, q_{acc}, q_{rej})$, компоненты которой имеют следующий вид.

- Конечное множество Σ - *входной алфавит.*
- Другое конечное множество Γ - *рабочий алфавит*, содержащий все символы, допустимые на ленте, причём $\Sigma \subset \Gamma$. Рабочий алфавит содержит особый символ - пробел: $\_ \in \Gamma$, $\_ \notin \Sigma$.
- Конечное множество Q - множество *состояний.*
- Есть *начальное состояние* $q_0 \in Q$.
- *Функция переходов* $\delta : (Q \backslash \{q_{acc}, q_{rej}\}) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ определяет поведение машины на каждом шаге. Если машина находится в состоянии $q \in Q$ и обозревает символ $a \in \Gamma$, то $\delta(q,a)$ - это тройка (q', a', d), где $q' \in Q$ - новое состояние, a' - символ, записываемый на ленте вместо a, и $d \in \{-1, +1\}$ направление перемещения головки.
- Если машина переходит в *принимающее состояние* $q_{acc} \in Q$ или в *отвергающее состояние* $q_{rej} \in Q$, то она останавливается.

На данной входной строке $w \in \Sigma^*$ машина начинает своё вычисление в *начальной конфигурации* $q_0 w$, то есть, в состоянии q_0 и с символами w на ленте, окружённые пробелами в обоих направлениях $(\ldots \_\_w\_\_ \ldots)$. При этом головка смотрит на первый символ входной строки.

Конфигурация машины Тьюринга - это строка вида $\alpha q a \beta$, где $\alpha, \beta \in \Gamma^*$, $a \in \Gamma$ и $q \in Q$, означающая, что машина находится в состоянии q, головка обозревает указанный символ a, и на ленте записаны символы $\alpha a \beta$, окружённые бесконечным числом пробелов в обоих направлениях.

На каждом шаге, если машина находится в конфигурации $\alpha q a \beta$, то её конфигурация на следующем шаге однозначно определена в соответствии со значением функции переходов для текущего состояния $q \in Q$ и текущего символа $a \in \Gamma$. Если головка едет налево, следующая конфигурация имеет вид $\alpha q' b a' \beta$.

Однозначно определяется конечная или бесконечная последовательность конфигураций, называемая *вычислением* машины на строке w. Вычисление может или *остановиться* на некотором шаге, в том смысле, что машина перейдёт в принимающее или отвергающее состояние, или же оно может продолжаться бесконечно, в каковом случае говорится, что машина *зацикливается.* Строка *принимается* машиной Тьюринга M, если машина останавливается на ней в принимающем состоянии. Множество, распознаваемое машиной $(L(M))$ - это множество всех строк, которые она принимает.

Определение 2. *Запись машины Тьюринга* M - это строка $\sigma(M) \in \{0,1\}^*$, определяемая следующим образом. Пусть множество состояний - $Q = \{q_1, \ldots, q_n\}$ (если состояния назывались как-то иначе, их можно переименовать без ущерба для распознаваемого множества). Пусть входной алфавит - $\Sigma = \{a_1, \ldots, a_l\}$, рабочий алфавит - $\Gamma = \{a_1, \ldots, a_l, a_{l+1}, \ldots, a_m\}$, последний символ - пробел, начальное состояние - q_1, принимающее состояние - $q_{acc} = q_n$, отвергающее состояние - $q_{rej} = q_{n-1}$. Тогда машина кодируется следующим образом, где символ произведения означает несколько строк, записанных одна за одной.

$$\sigma(M) = 1^l 0 1^m 0 1^n 0 \Bigg(\prod_{\delta(q_i,a_j)=(q_{i'},a_{j'},d)} 1^i 0 1^j 0 1^k 0 1^r 0 1^{d+2} 0 \Bigg)$$

Рассмотрим теперь два языка: L_0 - записи машин Тьюринга, которые не принимаются самой собой. L_1 - наоборот, записи, которые машины сами примут. Из очевидных противоречий можно получить следующую теорему:

Теорема 1. *Множество L_0 не распознаётся никакой машиной Тьюринга.*

А из неё можно получить следующий факт:

Теорема 2. *Множество $L_\varnothing = \{\sigma(M) | L(M) = \varnothing\}$ записей машин Тьюринга, распознающих пустой язык, не распознаётся никакой машиной Тьюринга.*

Доказательство. Пусть $L_\varnothing$ распознаётся некоторой машиной Тьюринга $M_\varnothing$. Тогда, используя эту машину в качестве подпрограммы, предполагается построить новую машину M_0, которая будет распознавать множество L_0. Эта машина M_0 работает следующим образом.

- На входе M_0 получает какую-то строку вида $\sigma(M)$, где M - машина Тьюринга.
- Далее, используя эту строку $\sigma(M)$, машина M_0 строит у себя на ленте запись новой машины M' в виде строки $\sigma(M')$. Эта новая машина работает так.

  (a) Получив на входе некоторую строку w, машина M' стирает эту строку и записывает вместо неё фиксированную строку $\sigma(M)$.

  (b) Далее M' работает в точности как M.

  Чтобы построить M', достаточно приделать к M новое начальное состояние и несколько состояний, в которых w сотрёт содержимое ленты и запишет вместо него $\sigma(M)$; все остальные переходы останутся такими же, как были в M.

- Наконец, машина M_0 запускает машину $M_\varnothing$ на входе $\sigma(M')$, чтобы проверить, пустой ли язык распознаёт M'. Если M' принимает, то M_0 тоже принимает.

По построению, машина M' примет любую входную строку, если M принимает $\sigma(M)$; если же M не принимает $\sigma(M)$, то M' не примет ничего. Таким образом, если $\sigma(M) \notin L(M)$, то $L(M') = \varnothing$,а если $\sigma(M) \in L(M)$,то $L(M') = \Sigma^* \neq \varnothing$.

Следовательно, $L(M') = \varnothing$ тогда и только тогда, когда $\sigma(M) \notin L(M)$. Поэтому M_0 распознаёт L_0, что невозможно по *теореме 1.* Полученное противоречие означает, что сделанное вначале предположение о существовании машины Тьюринга $M_\varnothing$ неверно. □

Определение 3. Задача *разрешима*, если существует машина Тьюринга, останавливающаяся на любом входе и выдающая правильный ответ. *Неразрешимая* - иначе.

Билет 2.

Примеры неразрешимых задач. Теорема Райса.

Лекция 1. Введение в предмет ТИ: машины Тьюринга, неразрешимые задачи.

Конспект, запись.

Лекция 2. Теорема Райса. Формальные языки. Детерминированные конечные автоматы и их ограничения. Регулярная лемма о накачке. Недетерминированные конечные автоматы.

Конспект, запись.

Теорема 3. *Множество L_1 неразрешимо, то есть, не распознаётся никакой машиной Тьюринга, останавливающейся на любом входе.*

Доказательство. Пусть решается некоторой машиной M_1. Тогда можно построить новую машину Тьюринга M_0, которая всю дорогу работает в точности как M_1, но когда последняя соберётся принимать, M_0 отвергнет; если же моделируемая M_1 решит отвергнуть, то M_0 примет. Поскольку M_1 останавливается на любом входе, оказывается, что M_0 распознаёт язык L_0. Это, однако, невозможно по **теореме 1**. $\square$

Теорема 4. *Множество L_ε неразрешимо, то есть не распознаётся никакой машиной Тьюринга, останавливающейся на любом входе ($L_\varepsilon = \{\sigma(M) | \varepsilon \in L(M)\}$).*

Доказательство. Пусть L_ε решается некоторой машиной Тьюринга M_ε. Тогда, используя эту машину в качестве подпрограммы, предполагается построить новую машину Тьюринга M_1, которая будет распознавать множество L_1. Эта машина M_1 работает следующим образом.

- На входе M_1 получает строку $\sigma(M)$.
- Далее, используя эту строку $\sigma(M)$, машина M_1 строит у себя на ленте запись новой машины M' в виде строки $\sigma(M')$. Эта новая машина работает так.

  (a) Получив на входе некоторую строку w, машина M' немедленно её стирает, записывая себе на ленту вместо w фиксированную строку $\sigma(M)$.

  (b) Далее M' работает в точности как M.

  Чтобы построить M', достаточно приделать к M новое начальное состояние и несколько состояний, в которых w будет заменено на $\sigma(M)$; все остальные переходы останутся такими же, как были в M.
- Наконец, машина M_1 запускает машину M_ε на входе $\sigma(M')$, чтобы проверить, принадлежит ли $\sigma(M')$ языку L_ε. Машина M_1 принимает, если $\sigma(M') \in L_\varepsilon$, и отвергает, если $\sigma(M') \notin L\varepsilon$.

По построению, машина M' будет работать так: какую бы строку ей ни дали на входе, она примет её тогда и только тогда, когда M принимает $\sigma(M)$. Таким образом, если $\sigma(M) \in L(M)$, то $L(M') = \sigma^*$, а если $\sigma(M) \notin L(M)$, то $L(M') = \varnothing$.

Следовательно, $\varepsilon \in L(M')$ тогда и только тогда, когда $\sigma(M) \in L(M)$. Поэтому M_1 распознаёт L_1 и останавливается на любом входе, что невозможно по **теореме 3**. Полученное противоречие означает, что сделанное вначале предположение о существовании машины Тьюринга M_ε неверно. □

Теорема 5 (*Райс*)**.** *Всякое нетривиальное свойство языка, распознаваемого данной на входе машиной Тьюринга, неразрешимо.*

Доказательство. Пусть $\mathcal{L}$ - нетривиальное свойство языков, распознаваемых машинами Тьюринга. Разрешить это свойство - это значит построить машину Тьюринга, останавливающуюся на любом входе и распознающую язык $L_\mathcal{L} = \{\sigma(M)|L(M) \in L\}$. Пусть такая машина $M_\mathcal{L}$ существует, то есть, $L(M_\mathcal{L}) = L_\mathcal{L}$.

Сперва, пусть $\varnothing \notin \mathcal{L}$. Поскольку свойство нетривиально, существует язык, этим свойством обладающий и распознаваемый некоторой машиной Тьюринга: $\widehat{L} \in \mathcal{L}$, где $\widehat{L} = L(\widehat{M})$. Тогда строится следующая машина Тьюринга M_1, распознающая множество $L_1 = \{\sigma(M)|\sigma(M) \in L(M)\}$ и останавливающаяся на любом входе.

- На входе: $\sigma(M)$.
- Построить новую машину $\widetilde{M}$, работающую так:

  (a) Получив на входе строку w, машина $\widetilde{M}$ сохраняет её про запас.

  (b) Далее, $\widetilde{M}$ запускает M на $\sigma(M)$. Если M отвергает, то $\widetilde{M}$ тоже отвергнет, если M зацикливается, то $\widetilde{M}$ зациклится с нею, а если M примет - $\widetilde{M}$ работает дальше.

  (c) Наконец, $\widetilde{M}$ очищает всю ленту, записывает на неё строку w и передаёт управление машине $\widehat{M}$.
- Запустив $M_\mathcal{L}$ на входе $\sigma(\widetilde{M})$, проверить, принадлежит ли $L(\widetilde{M})$ множеству $\mathcal{L}$. Если принадлежит - принять, а если нет, то отвергнуть.

По построению, машина $\widetilde{M}$ будет работать так: если $\sigma(M) \in L(M)$, то она работает в точности как $\widehat{M}$, и потому $L(\widetilde{M}) = L(\widehat{M})$; поскольку $\widehat{L} \in \mathcal{L}$, в этом случае машина $M_\mathcal{L}$ даст ответ «да». Если же $\sigma(M) \notin L(M)$, то $L(\widetilde{M}) = \varnothing \notin \mathcal{L}$, и машина $M_\mathcal{L}$ даст ответ «нет». Стало быть, $L(M_1) = L_1$, и при этом M_1 останавливается на любом входе - а это невозможно по **теореме 3**.

Случай $\varnothing \in \mathcal{L}$ сводится к предыдущему так: если свойство $\mathcal{L}$ разрешимо, то разрешимо и его отрицание, то есть, язык $\{\sigma(M)|L(M) \in 2^{\Sigma^*} \setminus \mathcal{L}\}$. А неразрешимость этого свойства доказана выше. □

Билет 3.

Конечные автоматы: DFA, NFA, их равномощность. Несуществование конечного автомата для языка $\{a^n b^n | n \geq 0\}$.

Лекция 2. Теорема Райса. Формальные языки. Детерминированные конечные автоматы и их ограничения. Регулярная лемма о накачке. Недетерминированные конечные автоматы.

Конспект, запись.

Определение 4. *Детерминированный конечный автомат* (DFA) - пятёрка $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$, со следующим значением компонентов.

- Σ - алфавит (конечное множество).
- Q - конечное множество состояний.
- $q_0 \in Q$ - начальное состояние.
- Функция переходов $\delta : Q \times \Sigma \to Q$. Если автомат находится в состоянии $q \in Q$ и читает символ $a \in \Sigma$, то его следующее состояние - $\delta(q,a)$. Функция переходов определена для всех q и a.
- Множество принимающих состояний $F \subseteq Q$.

Для всякой входной строки $w = a_1 \ldots a_l$, где $l \geq 0$ и $a_1, \ldots, a_l \in \Sigma$, вычисление - последовательность состояний $p_0, p_1, \ldots, p_{l-1}, p_l$, где $p_0 = q_0$, и всякое следующее состояние p_i, где $i \in \{1, \ldots, l\}$, однозначно определено как $p_i = \delta(p_{i-1}, a_i)$.

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots \xrightarrow{a_l} p_l$$

Строка *принимается*, если последнее состояние p_l принадлежит множеству F - иначе *отвергается*.

Язык, распознаваемый автоматом, обозначаемый через $L(\mathcal{A})$ - это множество всех строк, которые он принимает.

Определение 5. *Недетерминированный конечный автомат* (NFA) - пятёрка $\mathcal{B} = (\Sigma, Q, Q_0, \delta, F)$, со следующим значением компонентов.

- Σ - алфавит (конечное множество).
- Q - конечное множество состояний.
- $Q_0 \subseteq Q$ - начальное состояние.
- Функция переходов $\delta : Q \times \Sigma \to 2^Q$. Если автомат находится в состоянии $q \in Q$ и читает символ $a \in \Sigma$, то его следующее состояние - любое из $\delta(q,a)$.
- Множество принимающих состояний $F \subseteq Q$.

Для всякой входной строки $w = a_1 \dots a_l$, вычисление - последовательность состояний $p_0, p_1, \dots, p_{l-1}, p_l$, где $p_0 \in Q_0$, и всякое следующее состояние $p_i \in \delta(p_{i-1}, a_i)$, для $i \in \{1, \dots, l\}$.

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \dots \xrightarrow{a_l} p_l$$

Строка *принимается*, если последнее состояние p_l принадлежит множеству F - иначе *отвергается*.

Язык, распознаваемый автоматом, обозначаемый через $L(\mathcal{B})$ - это множество всех строк, которые он принимает.

Лемма 1 (Построение подмножеств)**.** *Пусть $\mathcal{B} = (\Sigma, Q, Q_0, \delta, F)$ - произвольный NFA. Тогда существует DFA $\mathcal{A} = (\Sigma, 2^Q, Q_0, \delta', F')$, состояния которого - подмножества Q, который распознаёт тот же язык, что и $\mathcal{B}$. Его переход в каждом состоянии-подмножестве $S \subseteq Q$ по каждому символу $a \in \Sigma$ ведёт во множество состояний, достижимых по a из некоторого состояния их S.*

$$\delta'(S,a) = \bigcup_{q \in S} \delta(q,a)$$

Состояние-подмножество $S \subseteq Q$ - принимающее, если оно содержит хотя бы одно принимающее состояние NFA.

$$F' = \{S | S \subseteq Q, S \cap F \neq \varnothing\}$$

Доказательство. Более подробно стоит почитать в конспекте Охотина. Постараюсь собрать здесь краткий пересказ. □

Утверждение 1. Язык $L = \{a^n b^n | n \geq 0\}$ не распознаётся никаким конечным автоматом.

Доказательство. От противного: пусть он распознаётся некоторым DFA

$$\mathcal{A} = (\{a,b\}, Q, q_0, \delta, F)$$

Для всякой строки a_i, где $i \geq 0$, пусть $q_i = \delta(q_0, a_i)$ - состояние DFA по прочтении этой строки.

Пусть $n = |Q|$. Тогда, читая первую половину строки $w = a^n b^n$, автомат пройдёт через последовательность из $n+1$ состояний $p_0, \dots, p_n$, а после, читая вторую половину, ещё через состояния $p_{n+1}, \dots, p_{2n}$. Так как строка принадлежит языку L, последнее состояние должно быть принимающим: $p_{2n} \in F$.

Поскольку всего различных состояний n, в последовательности $p_0, \dots, p_n$ будет пара одинаковых состояний: то есть, существуют числа i и j, где $0 \leq i < j \leq n$, для которых p_i совпадает с p_j. Тогда вычисление на строке $w' = a^{n-(j-i)} b^n$ имеет вид

$$p_0, \dots, p_i, p_{j+1}, \dots, p_n, \dots, p_{2n}$$

то есть, автомат не замечает, что из строки вырезали кусок. Получается, что автомат принимает строку w', которая не лежит в L - противоречие. □

Билет 4.

Формальные языки и действия над ними. Регулярные выражения, их равномощность конечным автоматам.

Лекция 2. Теорема Райса. Формальные языки. Детерминированные конечные автоматы и их ограничения. Регулярная лемма о накачке. Недетерминированные конечные автоматы.

Конспект, запись.

Лекция 3. Регулярные выражения. Преобразование регулярных выражений в автоматы. Преобразование автоматов в регулярные выражения...

Конспект, запись.

Определение 6. *Формальный язык* - подмножество множества конечных слов над конечным алфавитом.

Их можно *объединять*, *пересекать*, брать *дополнение* до всего множества конечных слов. Также существует *конкатенация* двух языков - язык, состоящий из конкатенаций всех пар слов из двух двнных. Если же мы хотим *повторить* сколько-то раз слова из одного языка (не обязательно одно и то же), то это будет обозначаться как L^k, где k - количество слов в конкатенации. L^* - *звёздочка Клини* - объединение всех таких множество по $k \in \mathbb{Z}_+$.

Определение 7. Регулярные выражения над алфавитом Σ определяются так.

- Символ для пустого множества $\varnothing$ - регулярное выражение.
- Всякий символ a, где $a \in \Sigma$ - регулярное выражение.
- Если α и β - регулярные выражения, то тогда $(\alpha|\beta)$, $(\alpha\beta)$и$(\alpha)^*$ - тоже регулярные выражения.

Всякое регуларное выражение α определяет язык над алфавитом Σ, обозначаемый через $L(\alpha)$. Символ для пустого множества определяет пустое множество.

$$L(\varnothing) = \varnothing$$

Всякий символ из Σ обозначает одноэлементное множество, состоящее из односимвольной строки.

$$L(a) = \{a\}$$

Оператор выбора задаёт объединение множеств.

$$L(\alpha|\beta) = L(\alpha) \cup L(\beta)$$

Конкатенация регулярных выражений задаёт конкатенацию языков. Оператор итерации задаёт итерацию.

$$L(\alpha\beta) = L(\alpha) \cdot L(\beta)$$
$$L(\alpha^*) = L(\alpha)^*$$

Примечание 1. Преобразование автоматов в выражения и наоборот - в конспекте Охотина, объёмный материал, который, впрочем, стоит просто понять, как работает.

Билет 5.

Лемма о накачке для регулярных языков. Нерегулярный язык, удовлетворяющий лемме о накачке. Замкнутость класса регулярных языков относительно булевых операций, конкатенации, итерации, поэлементного квадратного корня.

Лекция 3. Регулярные выражения. Преобразование регулярных выражений в автоматы. Преобразование автоматов в регулярные выражения...

Конспект, запись.

Лемма 2 (*О накачке*)**.** *Для всякого регулярного языка $L \subseteq \Sigma^*$ существует такая константа $p \geq 1$, что для всякой строки $w \in L$ длины не менее чем p, существует разложение $w = xyz$, где y непусто, $|xy| \leq p$, и $xy^k z \in L$ для всех $k \geq 0$.*

Доказательство. Пусть $A = (\Sigma, Q, q_0, \delta, F)$ - DFA, распознающий L, и пусть $p = |Q|$. Для произвольной строки $w \in L$, удовлетворяющей $|w| \geq p$, пусть $r_0, r_1, \ldots, r_{|w|} \in Q$ - вычисление A на w, то есть, всякое состояние r_i достигается по прочтении первых i символов w. Рассматривается начальный участок этой последовательности, $r_0, r_1, \ldots, r_p$. Так как этот участок содержит более чем $|Q|$ состояний, среди них есть пара одинаковых, $r_i = r_j$, где $0 \leq i < j \leq p$.

Обозначая первые i символов w через x, последние $|w|-j$ символов w через z, и средний участок w через y, получается разложение $w = xyz$, где $|y| = j - i$. Пусть $q = r_i = r_j$. Тогда $\delta(q_0,x) = q$, $\delta(q,y) = q$ и $\delta(q,z) = r_{|w|} \in F$. Всякая строка вида $xy^k z$ тогда принимается автоматом A, поскольку он посещает состояние q по прочтении каждого префикса вида xy^k. $\square$

Утверждение 2. Язык $L = \{(ab)^n a^n | n \geq 1\} \cup (\{a,b\}^* \setminus (ab)^+ a^+)$ нерегулярен, однако он удовлетворяет условию леммы о накачке с константой $p = 3$.

Доказательство. Для доказательства нерегулярности можно воспользоваться замкнутостью относительно пересечения. Пусть L регулярен. Тогда его пересечение с языком $(ab)^+ a^+$ также должно быть регулярно. Однако, это пересечение - это язык $\{(ab)^n a^n | n \geq 1\}$, который не удовлетворяет лемме о накачке и потому оказывается нерегулярным. Полученное противоречие доказывает нерегулярность языка L.

Чтобы проверить условие леммы о накачке с константой $p = 3$, для всякой строки из L длины не менее чем 3 необходимо построить её разложение вида xyz, для которого все строки вида $xy^i z$ принадлежат L. Разложение определяется в зависимости от первых трёх символов строки - так, чтобы ни одна из накачанных строк $xy^i z$ не попала в $(ab)^+ a^+$.

- Строка $abw \in L$, где $w \in \{a,b\}^*$, разбивается на $x = \varepsilon$, $y = a$, $z = bw$: тогда строка, полученная после накачки, начинается или с b, или с aa.
- Строка $bbw \in L$ разбивается на $x = \varepsilon$, $y = b$, $z = bw$: тогда после накачки она будет начинаться с b.
- Строка $satw \in L$, где $s,t \in \{a,b\}$ - её первый и третий символы, разбивается на $x = sa$, $y = t$, $z = w$: накачанная строка начинается с sa, и потому, как и во всех предыдущих случаях, принадлежит L.

Стало быть, искомое разложение xyz существует для любой строки из L длины хотя бы 3, и условие леммы о накачке выполняется. $\square$

Замкнутость класса регулярных языков относительно булевых операций очевидна, так как им соответствуют детерминированные конечные автоматы, у которых можно взять прямое произведение, то есть, составить матрицу пар состояний и ходить по ним. Что касается корня.

Теорема 6. *Для всякого DFA* $A = (\Sigma, Q, q_0, \delta, F)$ *поэлементный квадратный корень* $\sqrt{L(A)}$ *распознаётся DFA* B *со множеством состояний* $Q^Q = \{f | f : Q \to Q\}$.

Доказательство. Для каждой строки $w \in \Sigma^*$, если A начинает своё вычисление на w в состоянии $q \in Q$, то пусть состояние, в котором он завершает чтение w, обозначается через $f_w(q)$. Тогда f_w - это функция $f_w : Q \to Q$, называемая *поведением* A на w.

Лемма 3. *Для всякого DFA* $A = (\Sigma, Q, q_0, \delta, F)$ *существует DFA* B *со множеством состояний* $Q^Q = \{f | f : Q \to Q\}$, *вычисляющий поведение* A *на прочитанной строке.*

Доказательство. Начальное состояние B - тождественная функция на множестве Q, то есть, поведение A на пустой строке.

Чтобы определить переходы, вводится обозначение $\delta_a : Q \to Q$ для *функции переходов* A по каждому символу $a \in \Sigma$ - то есть, $\delta_a(q) = \delta(q,a)$. Тогда поведение A на строке wa, где $w \in \Sigma^*$ и $a \in \Sigma$ - это композиция поведения на w и функции переходов по a. Тогда B вычисляет эту композицию на каждом шаге: $\delta'(f, a) = \delta_a \circ f$. □

Новый автомат B, будучи запущенным на w, вычисляет поведение A на w. После этого достаточно определить множество принимающих состояний как $F' = \{f | f(f(q_0)) \in F\}$.

□

Билет 6.

Двухсторонние автоматы (2DFA), их перевод в детерминированные односторонние(DFA).

Лекция 4. Минимальные DFA. Алгоритм минимизации DFA. Двухсторонние конечные автоматы (2DFA). Преобразование 2DFA к DFA.

Конспект, запись.

Определение 8. *Двухсторонний детерминированный конечный автомат* (2DFA) - это пятёрка $A = (\Sigma, Q, q_0, \delta, F)$, состоящая из следующих компонентов.

- Σ - алфавит, не содержащий двух особых символов $\vdash, \dashv \notin \Sigma$ (меток начала и конца строки).
- Q - конечное множество состояний.
- $q_0 \in Q$ - начальное состояние.
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to Q \times \{-1, +1\}$ - частичная функция переходов.
- $F \subseteq Q$ - множество *принимающих состояний*, применяемых на правой метке конца ($\dashv$).

Для входной строки $w = a_1 \dots a_l \in \Sigma^*$, пусть $a_0 =\vdash$ и $a_{l+1} =\dashv$. Вычисление 2DFA на w - это самая длинная, возможно бесконечная, последовательность $(p_0,i_0), \dots, (p_j,i_j), \dots$, где:

- $p_j \in Q$ и $0 \le i_j \le l + 1$ на каждом j-м шаге;
- начальная конфигурация - $(p_0, i_0) = (q_0, 0)$;
- всякая следующая конфигурация (p_j, i_j), если она определена, удовлетворяет $\delta(p_j - 1, a_{i_{j-1}}) = (p_j, d_j)$ и $i_j = i_{j-1} + d_j$.

Вычисление всегда определено однозначно. Строка w *принимается*, если вычисление конечно и его последняя конфигурация - $(p_f, l + 1)$, для некоторого $p_f \in F$. Также вычисление может заканчиваться неопределённым переходом (ведь δ - частичная функция) или же продолжаться бесконечно; и в том и в другом случае A не принимает w.

Язык, распознаваемый автоматом - $L(A) = \{w|A \text{ принимает } w\}$.

Теорема 7. *Для всякого 2DFA существует DFA, распознающий тот же язык.*

Лемма 4. *Для всякого 2DFA с n состояниями существует DFA с $n(n^n - (n-1)^n) + 1$ состояниями, распознающий тот же язык.*

Что касается доказательства всего этого, оно занимает три страницы конспекта (9-12), ссылка в шапке билета.

Билет 7.

Минимальные DFA. Алгоритм минимизации DFA.

Лекция 4. Минимальные DFA. Алгоритм минимизации DFA. Двухсторонние конечные автоматы (2DFA). Преобразование 2DFA к DFA.

Конспект, запись.

Лемма 5. *Пусть в DFA* $A = (\Sigma, Q, q_0, \delta, F)$*, начиная из двух состояний,* q *и* q'*, принимается одно и то же множество строк:* $L_A(q) = L_A(q')$*, где* $q \neq q'$*. Пусть* $q' \neq q_0$*. Тогда автомат* $B = (\Sigma,Q\backslash\{q'\},q_0,\delta',F\backslash\{q'\})$*, полученный удалением* q' *и перенаправлением всех переходов, ведущих в* q'*, в состояние* q*, распознаёт тот же язык.*

Доказательство. Доказывается следующее утверждение: для всякой строки $v \in \Sigma^*$ и для всякого состояния $p \in Q\backslash\{q'\}$, строка v лежит в $L_A(q)$ тогда и только тогда, когда она лежит в $L_B(q)$. Индукция по длине v.

Базис $v = \varepsilon$. Верно, так как p - принимающее в A тогда и только тогда, когда оно принимающее в B.

Индукционный переход от v к av, где $a \in \Sigma$. Пусть для строки v утверждение доказано, рассматривается строка av.

Если автомат A читает av из состояния p и переходит по a в состояние $r \neq q'$, то тогда B тоже переходит по a в r. По предположению индукции, $v \in L_A(r)$ тогда и только тогда, когда $v \in L_B(r)$, и потому вычисления A и B из p по av тоже имеют одинаковый исход.

Если же A переходит по a в состояние q', то B переходит в q. В этом случае известна следующая цепочка равносильностей: $v \in L_A(q')$ тогда и только тогда, когда $v \in L_A(q)$, что в свою очередь верно тогда и только тогда, когда $v \in L_B(q)$. Отсюда снова следует одинаковый исход вычислений из p по av. □

Лемма 6. *Пусть* $L \subseteq \Sigma^*$ *- регулярный язык. Если для двух строк* $u, v \in \Sigma^*$ *существует такая строка* $w \in \Sigma^*$*, что одна из строк* uw*,* vw *принадлежит языку* L*, а другая не принадлежит, то во всяком DFA, распознающем* L*, состояния* $\delta(q_0,u)$ *и* $\delta(q_0,v)$ *должны быть различны.*

Доказательство. Если такие два состояния совпадут, то автомат или примет обе строки uw, vw, или обе отвергнет. □

Теорема 8. *Минимальный DFA для данного регулярного языка единственен с точностью до изоморфизма.*

Доказательство. Пусть $A = (\Sigma, P, p_0, \eta, E)$ и $B = (\Sigma, Q, q_0, \delta, F)$ - два минимальных DFA, распознающие один и тот же язык L. Нужно построить взаимно однозначное соответствие между P и Q, сохраняющее переходы.

Для всякого состояния первого автомата, $p \in P$, пусть up - кратчайшая строка, по которой оно достижимо (если состояние недостижимо, его можно удалить, и тогда автомат не был бы минимальным). Тогда из состояния p принимается язык $L_A(p) = \{v | u_p v \in L\}$.

В силу минимальности A, никакие два из этих языков не совпадают - иначе соответствующие состояния можно было бы объединить по **лемме ?**.

Читая эту же строку u_p, автомат B приходит в некоторое состояние q_p, из которого принимается тот же самый язык $L_B(q_p) = \{v | u_p v \in L\}$. Эти языки также попарно не совпадают. Тогда состоянию p ставится в соответствие состояние q_p, и, в частности, p_0 отображается в q_0.

Поскольку у B столько же состояний, сколько и у A, никаких других состояний у него нет, и получено взаимно однозначное соответствие между P и Q. Это соответствие сохраняет переходы: а именно, если p соответствует q, то состояния $p' = \eta(p,a)$ и $q' = \delta(q,a)$ также должны соответствовать друг другу. Действительно, из каждого из этих состояний должен приниматься язык $\{v | u_p a v \in L\}$, и в каждом из автоматов есть только одно такое состояние. □

[сюда вставить алгоритм]

Теорема 9. *Алгоритм 1 строит автомат, распознающий тот же язык, что и исходный, и минимальный среди всех автоматов, распознающих этот язык. Алгоритм работает за время $O(|\Sigma| \cdot n^2)$, где n - число состояний в исходном автомате, а Σ - алфавит.*

Доказательство. 4-5 страница конспекта по ссылке в шапке билета. □

Билет 8.

Вероятностные автоматы (PFA). Теорема Майхилла–Нероуда о классах эквивалентности строк.

Лекция 5. Теорема Майхилла–Нероуда. Вероятностные конечные автоматы (PFA). Двухсторонние вероятностные конечные автоматы (2PFA). Введение в формальные грамматики.

Конспект, запись.

Определение 9. *Вероятностный конечный автомат* (PFA) - это пятёрка $B = (\Sigma, Q, q_0, \delta, F)$, со следующим значением компонентов.

- Σ - алфавит.
- Q - конечное множество состояний.
- $q_0 \in Q$ - начальное состояние.
- Функция переходов $\delta : Q \times \Sigma \to [0,1]^Q$, где $[0,1]^Q$ - это множество стохастических функций f из Q в $[0,1]$, для которых верно $\Sigma_{q \in Q} f(q) = 1$. Эта функция даёт вероятность перехода в каждое состояние. Находясь в состоянии $q \in Q$ и читая символ $a \in \Sigma$, автомат переходит во всякое состояние r с вероятностью $\delta(q,a)(r)$.
- Множество *принимающих состояний* $F \subseteq Q$.

Вычисление на входной строке $w = a_1 \ldots a_l$ - это всякая последовательность состояний p_0,p_1, ... ,p_{l-1},p_l.

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots \xrightarrow{a_l} p_l$$

Такое вычисление называется вычислением из p_0 в $p + l$. Вероятность вычисления - произведение $\prod_{i=1}^{l} \delta(p_{i-1}, a_i)(p_i)$.

Вероятность придти из p в q, прочитав строку w - это сумма вероятностей всех таких вычислений. Вероятность принять из данного состояния p - это сумма вероятностей придти из p во все принимающие состояния. Вероятность принять строку - это вероятность принять её из начального состояния.

Автомат должен удовлетворять следующему условию ограниченной ошибки: всякая строка принимается или с вероятностью хотя бы $\frac{2}{3}$, или с вероятностью не более чем $\frac{1}{3}$. Если строка принимается с вероятностью хотя бы $\frac{2}{3}$, то считается, что она принимается, а если с вероятностью $\frac{1}{3}$ или менее - то что отвергается. Язык, распознаваемый автоматом, обозначаемый через $L(B)$ - это множество всех входных строк, которые принимаются с вероятностью более чем $\frac{1}{2}$ (что равносильно тому, что она принимается с вероятностью хотя бы $\frac{2}{3}$).

Почему бы не упомянуть в билете ещё и следующую теорему...

Теорема 10. *Всякий PFA распознаёт регулярный язык.*

Теорема 11 (*Майхилл-Нероуд*). *Пусть L - язык, и пусть $\equiv_L$ - отношение эквивалентности на множестве строк, где $u \equiv_L u'$, если для всякой строки v, строки uv и $u'v$ или обе лежат в L или обе не лежат. Утверждается, что язык L регулярен тогда и только тогда, когда число классов эквивалентности в отношении $\equiv_L$ конечно.*

Доказательство. $\Rightarrow$ Классы эквивалентности - это состояния минимального DFA.

$\Leftarrow$ Обозначение: $[u]$ - класс эквивалентности, в который попала строка u.

Строится DFA, состояниями которого станут классы эквивалентности. Начальный класс - $q_0 = [\varepsilon]$. Переход из класса $S \subseteq \Sigma^*$ по символу $a \in \Sigma$ определяется как $\delta(S,a) = [ua]$, где u - произвольная строка из S; ниже будет показано, что это определение не зависит от выбора строки u. Наконец, класс $S \subseteq \Sigma^*$ - принимающее состояние построенного DFA, если $S \subseteq L$.

Утверждается, что, прочитав строку u, построенный DFA приходит в класс S, содержащий эту строку. Доказательство - индукция по длине u. Базовый случай: начальное состояние, $[\varepsilon]$, содержит ε.

Переход: если автомат пришёл в класс S по строке $\widetilde{u}$, то известно, что $\widetilde{u} \in S$. Для вычисления перехода по a выбирается какая-то строка $u \in S$, и автомат переходит в состояние $[ua]$. Утверждается, что $[ua]$ и $[u'a]$ - это один и тот же класс. Нужно показать, что $uav \in L$ тогда и только тогда, когда $u'av \in L$. Действительно, поскольку $u, \widetilde{u} \in S$, для их продолжений строкой av искомая эквивалентность выполнена.

Наконец, пусть автомат прочитал всю входную строку w и пришёл в состояние S, содержащее w. Тогда, если $w \in L$, то $S \subseteq L$, и потому это состояние будет принимающим. Если же $w \notin L$, то $S \cap L = \varnothing$, и состояние, соответственно, окажется отвергающим. $\square$

Билет 9.

Двухсторонние вероятностные автоматы (2PFA).

Лекция 5. Теорема Майхилла–Нероуда. Вероятностные конечные автоматы (PFA). Двухсторонние вероятностные конечные автоматы (2PFA). Введение в формальные грамматики.

Конспект, запись.

По образцу определяются двухсторонние вероятностные конечные автоматы (2PFA) - однако их выразительная мощность неожиданно оказывается большей, чем у PFA и 2DFA: Фрейвальдс [1981] построил следующий 2PFA, распознающий нерегулярный язык. Это, наверное, всё, что можно сюда закинуть, кроме одного интересного утверждения. Остальное лучше прочитать и вникнуть в параграфе 3 конспекта по ссылке в шапке билета.

Утверждение 3. Существует 2PFA, распознающий язык $\{a^n b^n | n \geq 0\}$.

Билет 10.

Формальные грамматики: определения через деревья разбора, через логический выводи через перезапись строк. Грамматики для абстрактных языков и для конструкцийязыков программирования. Замкнутость класса языков, задаваемых грамматиками,относительно объединения, конкатенации, повторения и взятия префиксов или суф-фиксов.

Лекция 5. Теорема Майхилла–Нероуда. Вероятностные конечные автоматы (PFA). Двухсторонние вероятностные конечные автоматы (2PFA). Введение в формальные грамматики.

Конспект, запись.

Лекция 6. Определения грамматик. Примеры построения грамматик. Действия над языками, выразимые в грамматиках. Ограничения выразительной мощности грамматик: лемма о накачке, лемма Огдена.

Конспект, запись.

Определение 10. *(Формальная) грамматика* - это четвёрка $G = (\Sigma, N, R, S)$, состоящая из следующих компонентов.

- Конечное множество *символов* Σ - алфавит определяемого языка.
- Конечное множество N - это множество определяемых в грамматике свойств строк, которым всякая строка над алфавитом Σ обладает или не обладает. Обозначаются обычно буквами $A, B, C, \ldots$

  В информатике элементы N традиционно называют «нетерминальными символами» («нетерминалами») - отсюда буква N - поскольку пути в дереве разбора на них не заканчиваются. В лингвистике они называются синтаксическими категориями.

- Конечное множество R правил грамматики, каждое из которых описывает возможную структуру строк со свойством $A \in N$ в виде конкатенации $u_0 B_1 u_1 \ldots B_l u_l$, где $B_1, \ldots, B_l$ $(l \geq 0)$ - все нетерминальные символы, на которые ссылается правило, а любые символы, написанные между ними, образуют строки $u_0, u_1, \ldots, u_l$.

$$A \to u_0 B_1 u_1 \ldots B_l u_l \quad (A \in N, l \geq 0, B_1, \ldots, B_l \in N, u_0, u_1, \ldots, u_l \in \Sigma^*)$$

- Начальный символ $S \in N$, от «sentence», обозначает множество всех синтаксически правильных строк, определяемых в грамматике.

Иначе можно определить следующим образом.

Определение 11. Для грамматики $G = (\Sigma, N, R, S)$, высказывания имеют вид «строка w имеет свойство A», где $w \in \Sigma^*$ и $A \in N$, и обозначаются через $A(w)$.

Пусть $A \to u_0 B_1 u_1 \ldots B_l u_l$ - правило, в котором $B_1, \ldots, B_l \in N$, где $l \geq 0$ - это все нетерминальные символы, на которые оно ссылается, а $u_0, u_1, \ldots, u_l \in \Sigma^*$ - символы между ними. Это правило позволяет сделать следующий логический вывод, для любых строк $v_1, \ldots, v_l$, где над чертой - посылки, а под чертой - следствие.

$$\frac{B_1(v_1), \ldots ,B_l(v_l)}{A(u_0 v_1 u_1 \ldots v_l u_l)}(A \to u_0 B_1 u_1 \ldots B_l u_l) \quad (\text{for all } v_1, \ldots ,v_l \in \Sigma^*)$$

Вывод высказывания $A(u)$ - это последовательность таких шагов вывода, где в качестве посылок на каждом шаге используются ранее выведенные высказывания: $I_j \subseteq \{A_i(u_i) | i \in \{1, \ldots , j-1\}\}$, для всех j.

$$\frac{I_1}{A_1(u_1)}, \frac{I_2}{A_2(u_2)}, \ldots , \frac{I_{z-1}}{A_{z-1}(u_{z-1})}, \frac{I_z}{A(u)}$$

Если такой вывод существует, это обозначается через $\vdash A(u)$.

Тогда, для всех $A \in N$, определяется $L_G(A) = \{w | \vdash A(w)\}$. Язык, задаваемый грамматикой - $L(G) = L_G(S) = \{w | \vdash S(w)\}$.

И даже ещё одним образом.

Определение 12. Пусть $G = (\Sigma, N, R, S)$ - грамматика. Если $\eta A \theta$ - строка над совмещённым алфавитом символов и нетерминальных символов $\Sigma \cup N$ (где $A \in N$ и $\eta,\theta \in (\Sigma \cup N)^*$), и если $A \to \alpha$ - некоторое правило для A, тогда строка $\eta A \theta$ может быть *переписана в* $\eta \alpha \theta$ *за один шаг*, что обозначается следующим образом.

$$\eta A \theta \Rightarrow \eta \alpha \theta \quad (\text{для всех } A \to \alpha \in R \text{ and } \eta, \theta \in (\Sigma \cup N)^*)$$

Последовательность строк $\alpha_0,\alpha_1, \ldots ,\alpha_n$ над алфавитом $\Sigma \cup N$, где $n \geq 0$, называется цепочкой перезаписи, если всякая строка α_i может быть перезаписана за один шаг в строку α_{i+1}. Это обозначается так.

$$\alpha_0 \Leftrightarrow \alpha_1 \Leftrightarrow \ldots \Leftrightarrow \alpha_n$$

Тогда говорится, что α_0 перезаписывается в α_n за n шагов (обозначение: $\alpha_0 \Leftrightarrow^n \alpha_n$). Также вводятся обозначения для перезаписи за ноль и более шагов ($\alpha \Leftrightarrow^* \beta$), за один и более шагов ($\alpha \Leftrightarrow^+ \beta$), и за не более чем n шагов ($\alpha \Leftrightarrow^{\leq n} \beta$).

Язык, задаваемый грамматикой ($A \in N$).

$$L_G(A) = \{w | w \in \Sigma^*, A \Leftrightarrow^+ w\}$$
$$L(G) = L_G(S)$$

И последним.

Определение 13. Пусть $G = (\Sigma,N,R,S)$ - грамматика. Соответствующая система языковых уравнений такова (for all $A \in N$):

$$A = \underbrace{\bigcup_{A \to u_0 B_1 u_1 \ldots B_l u_l \in R} \{u_0\} \cdot B_1 \cdot \{u_1\} \cdot \ldots \cdot B_l \cdot \{u_l\}}_{\varphi_A}$$

Пусть наименьшее решение имеет вид $A = L_A$, для всех $A \in N$. Тогда $L(G)$ определяется как L_S.

Теорема 12. *Пусть $G = (\Sigma,N,R,S)$ - грамматика, как в* ***определении ?****. Для всякого нетерминального символа $A \in N$ и для всякой строки $w \in \Sigma^*$, следующие утверждения равносильны:*

(T). существует дерево разбора w из A;

(D). высказывание $A(w)$ выводимо ($\vdash A(w)$);

(R). A можно перезаписать в w за один и более шагов ($A \Leftrightarrow^{+} w$);

(E). w принадлежит A-компоненту наименьшего решения системы языковых уравнений ($w \in L_A$).

Теорема 13. *Для всякой грамматики G и для всякого регулярного языка K, язык $L(G) \cap K$ описывается некоторой грамматикой.*

Теорема 14. *Пусть $G = (\Sigma, N, R, S)$ - грамматика. Тогда существует грамматика, задающая язык $prefixes(L) = \{u | \exists v : uv \in L(G)\}$.*

Билет 11.

Лемма о накачке для грамматик. Несуществование грамматики для языка $\{a^n b^n c^n | n > 0\}$. Невозможность полного описания синтаксиса языка программирования грам-матикой. Незамкнутость класса языков, задаваемых грамматиками, относительно пересечения и дополнения.

Лекция 6. Определения грамматик. Примеры построения грамматик. Действия над языками, выразимые в грамматиках. Ограничения выразительной мощности грамматик: лемма о накачке, лемма Огдена.

Конспект, запись.

Лекция 7. Грамматики над односимвольным алфавитом. Невыразимые действия над грамматиками. Удаление пустых правил, удаление единичных правил, нормальный вид Хомского. Синтаксический анализ за кубическое время, построение дерева разбора.

Конспект, запись.

Лемма 7. *Для всякого языка $L \subseteq \Sigma^*$, задаваемого грамматикой, существует такое число $p \geq 1$, что для всякой строки $w \in L$ длины не менее чем p ($|w| \geq p$) существует разбиение $w = xuyvz$, где $|uv| > 0$ и $|uyv| \leq p$, для которого выполняется $xu^i yv^i z \in L$ для всех $i \geq 0$.*

Доказательство. Пусть $G = (\Sigma, N, R, S)$ - грамматика, задающая язык L. Пусть $m = \max A \to \alpha \in R |\alpha|$. Тогда число p определяется как $p = m^{|N|+1}$.

Пусть $w \in L$ - строка длины не менее чем p. Доказательство основано на анализе структуры дерева разбора w. Внутренняя вершина s в дереве называется точкой ветвления, если в этом месте листья разделяются не менее чем на две непустых группы; иными словами, дело не обстоит так, что у одного потомка s все листья, а у остальных ни одного.

В этом дереве строится путь из корня, на каждом шаге выбирается наибольшее поддерево данной вершины. Разделение гарантировано, коль скоро в текущем поддереве не менее чем m листьев. В каждой точке ветвления число листьев уменьшается не более чем в m раз, и потому, после прохождения l точек ветвления, в текущем поддереве останется не менее $\frac{|w|}{m^l}$ листьев. Поскольку в строке не менее чем $m^{|N|+1}$ символов, можно проделать не менее чем $|N| + 1$ нетривиальных разбиений, и потому путь будет содержать не менее чем $|N| + 1$ точек ветвления.

Среди нижних $|N| + 1$ точек ветвления на этом пути где-то повторится дважды некоторая метка $A \in N$. На отрезке между этими двумя экземплярами A какие-то поддеревья ответвляются направо, какие-то - налево. Пусть u - строка листьев в левых поддеревьях, а v - строка листьев в правых. Поскольку на этом пути есть не менее одной точки ветвления (верхнее A - одна из этих точек), хотя бы одна из строк u и v должна быть непустой.

Пусть $w = xuyvz$ - разбиение всей строки, в котором обозначены эти подстроки u и v. Такое разбиение показано на рис. (слева).

Участок дерева между двумя указанными экземплярами A можно повторить 0 и более раз, получая деревья разбора для строки $xu^i yv^i z$. Повторение 2 раза показано на рис.

(посередине), а повторение 0 раз - на рис. (справа).

[вставить рисунок] □

Утверждение 4. Язык $L = \{a^n b^n c^n | n \geq 0\}$ не задаётся никакой грамматикой.

Доказательство. Пусть есть, и пусть $p \geq 1$ - константа из леммы о накачке. Тогда для строки $w = a^p b^p c^p$ есть разбиение $w = xuyvz$, и нужно рассмотреть несколько случаев возможных разбиений.

- Если одна из строк u,v содержит символы разных типов, т.е., пересекает границу между a, b и c. Тогда $xu^2yv^2z \notin a^*b^*c^*$, и потому эта строка не может принадлежать L.
- Если $u,v \in a^*$, то $xu^0yv^0z = xyz = a^{p-|uv|}b^p c^p \notin L$. Случаи $u,v \in b^*$ и $u,v \in c^*$ такие же.
- Если $u \in a^*$ и $v \in b^*$, то $xu^0yv^0z = a^{p-|u|}b^{p-|v|}c^p$; эта строка не принадлежит L, поскольку $p - |u| \neq p$ или $p - |v| \neq p$. Случай $u \in b^*$ и $v \in c^*$ такой же.

В каждом случае получено противоречие. □

Утверждение 5. Следующая строка - правильная программа на языке C тогда и только тогда, когда $i = j = k$.

$$main()int\,\underbrace{x\dots x}_{i\geq 1};\underbrace{x\dots x}_{j\geq 1} = \underbrace{x\dots x}_{k\geq 1};$$

Отсюда можно вывести, что множество всех правильных программ на C не описывается грамматикой. Грамматиками описывают существенную часть синтаксиса языков программирования, но не весь синтаксис целиком.

Теорема 15. *Класс языков, задаваемых грамматиками, не замкнут относительно пересечения и дополнения.*

Доказательство. Языки $L_1 = \{a^i b^l c^l | i,l \geq 0\}$ и $L_2 = \{a^m b^m c^j | j,m \geq 0\}$, задаются следующими грамматиками.

$$\begin{array}{cc} S_1 \to aS_1|A & S_2 \to S_2c|B \\ A \to bAc|\varepsilon & B \to aBb|\varepsilon \end{array}$$

Если предположить, что семейство замкнуто относительно пересечения, то тогда пересечение двух вышеприведённых языков также будет описывается грамматикой.

$$L_1 \cap L2 = \{a^i b^l c^l | i,l \geq 0\} \cap \{a^m b^m c^j | j,m \geq 0\} = \{a^n b^n c^n | n \geq 0\}$$

Однако ранее было доказано, что грамматики для этого пересечения не существует - противоречие.

Незамкнутость относительно дополнения следует из этого результата, используя замкнутость относительно объединения. □

Билет 12.

Язык, удовлетворяющий лемме о накачке, но не задаваемый никакой грамматикой.Лемма Огдена - обобщение леммы о накачке. Несуществование грамматики для языка $\{a^l b^m c^n | l, m, n$ попарно

Лекция 6. Определения грамматик. Примеры построения грамматик. Действия над языками, выразимые в грамматиках. Ограничения выразительной мощности грамматик: лемма о накачке, лемма Огдена.

Конспект, запись.

Лекция 7. Грамматики над односимвольным алфавитом. Невыразимые действия над грамматиками. Удаление пустых правил, удаление единичных правил, нормальный вид Хомского. Синтаксический анализ за кубическое время, построение дерева разбора.

Конспект, запись.

Утверждение 6. Язык $\{a^l b^m c^n | l,m,n \geq 1, l \neq m, m \neq n, l \neq n\}$ удовлетворяет лемме о накачке с константой р $= 7$ (вместе с **утв. ?** даёт ответ на первое предложение).

Доказательство. Для всякой строки $w = a^{l_a} b^{l_b} c^{l_c} \in L$, где $|w| \geq 7$, пусть $s \in \{a, b, c\}$ - тот из символов, который встречается в w чаще, чем два других, так что l_s - наибольшее из чисел l_a, l_b и l_c. Тогда $l_s \geq 4$, поскольку если $l_s \leq 3$, то строка имела бы длину 6.

Пусть k - наименьшее натуральное число, отличающееся от $l_s - l_t$, для всех $t \in \{a, b, c\}$. Число k не превосходит 3, и потому меньше, чем l_s. Тогда предполагается накачивать блок из k символов s; иными словами, разбиение $w = xuyvz$, обещанное в лемме о накачке, определяется, полагая $u = s^k$ и $v = \varepsilon$, где x, y и z задаются подобающим образом, чтобы вышло $xuyvz = w$.

Для всякого $i \geq 1$, строка $xu^i yv^i z$ принадлежит L, поскольку количество символов s остаётся наибольшим, в то время как количество двух других символов не изменяется. Для $i = 0$, строка $xu^0 yv^0 z$, из которой удалена подстока s^k, всё ещё принадлежит языку L, потому что, согласно условию $l_s - l_t \neq k$, получившееся количество символов s отличается от количества каждого из оставшихся символов. $\square$

Лемма 8 (Огдена). *Для всякого языка $L \subseteq \Sigma^*$, задаваемого грамматикой, существует такое число $p \geq 1$, что для всякой строки $w \in L$ и для всякого множества $P \subseteq \{1, \ldots, |w|\}$ выделенных позиций в строке w, где $|P| \geq p$, существует разбиение $w = xuyvz$, где uv содержит хотя бы одну выделенную позицию, uyv содержит не более p выделенных позиций, и выполняется $xu^i yv^i z \in L$ для всех $i \geq 0$.*

Доказательство. Доказательство дословно повторяет доказательство леммы о накачке, с тем единственным отличием, что размер поддеревьев считается не по числу листьев, а по числу выделенных листьев. $\square$

Утверждение 7. Язык $L = \{a^l b^m c^n | l \neq m, m \neq n, l \neq n\}$ не удовлетворяет условию леммы Огдена и потому не задаётся никакой грамматикой.

Доказательство. Пусть p - число, данное леммой Огдена. Пусть $w = a^{p+p!} b^p c^{p+2p!}$ - строка, в которой выделены символы b^p. Лемма Огдена даёт разбиение $w = xuyvz$.

Если u или v содержит символы двух различных типов, то достаточно накачать дважды, чтобы получить строку xu^2uv^2z, не лежащую в $a^*b^*c^*$ и соответственно не попадающую в L.

Если как u, так и v попадают в b^p, то b^p накачивается $\frac{p!+1}{|uv|}$ раз до достижения $b^{p+p!}$, так что получается строка $a^{p+p!}b^{p+p!}c^{p+2p!}$, не принадлежащая языку L.

Если u попадает в $a^{p+p!}$ а v - в b^p, то они вместе накачиваются, пока b^p не достигает $b^{p+2p!}$. Что при этом происходит с $a^{p+p!}$ - неважно, поскольку всякая строка $a^k b^{p+2p!} c^{p+2p!}$, где $k \geq 0$, не принадлежит языку.

Если u попадает в b^p, а v попадает в $c^{p+2p!}$, то они накачиваются, пока b^p не превратится в $b^{p+p!}$. $\square$

Билет 13.

Конечные автоматы и грамматики над односимвольным алфавитом. Незамкнутость класса языков, задаваемых грамматиками, относительно деления.

Лекция 7. Грамматики над односимвольным алфавитом. Невыразимые действия над грамматиками. Удаление пустых правил, удаление единичных правил, нормальный вид Хомского. Синтаксический анализ за кубическое время, построение дерева разбора.

Конспект, запись.

Теорема 16. *Всякая грамматика над односимвольным алфавитом $\Sigma = \{a\}$ задаёт регулярный язык.*

Доказательство. По лемме о накачке.

Пусть $L \subseteq a^*$ - язык, задаваемый грамматикой. Тогда лемма о накачке ставит в соответсвие этому языку некоторую константу $p \geq 1$. Для всякой строки $a^n \in L$, где $n \geq p$, лемма о накачке утверждает, что для некоторого числа l, где $1 \leq l \leq p$, все строки вида $a^{n+i \cdot l}$, для всех $i \geq 0$, также лежат в L. В частности, раз $p!$ делится на l, все строки вида $a^{n+i \cdot p!}$, где $i \geq 1$, принадлежат L.

Для всякого остатка j по модулю $p!$, язык $L \cap a^{\geq p}$ или содержит какую-то строку длины j по модулю $p!$, или не содержит. Если он содержит хотя бы одну строку a^{n_j}, где $n_j \equiv j (\bmod p!)$, то он содержит и все более длинные такие строки. Поэтому язык представим так.

$$L = (L \cap a^{<p}) \cup \bigcup_{j \in \{0, \dots, p!-1\}, n_j \exists} a^{n_j} (a^{p!})^*$$

А это по существу регулярное выражение. □

Теорема 17. *Множество языков, задаваемых грамматиками, не замкнуто относительно операции поэлементного деления.*

Доказательство. Много схемок, поэтому, страница 3 конспекта. □

Билет 14.

Нормальный вид Хомского для грамматик: удаление пустых правил, удаление еди-ничных правил.

Лекция 7. Грамматики над односимвольным алфавитом. Невыразимые действия над грамматиками. Удаление пустых правил, удаление единичных правил, нормальный вид Хомского. Синтаксический анализ за кубическое время, построение дерева разбора.

Конспект, запись.

Следует ознакомиться с параграфом 3 конспекта по ссылке в шапке билета целиком.

Определение 14. Грамматика $G = (\Sigma, N, R, S)$ - в *нормальном виде Хомского*, если всякое правило из R имеет следующий вид.

$$\begin{array}{cc} A \to BC & (B,C \in N) \\ A \to a & (a \in \Sigma) \\ S \to \varepsilon & \end{array}$$

(последнее только если S не используется в правых частях никаких правил)

Теорема 18. *Для всякой грамматики можно построить грамматику в н.в.Хомского, задающую тот же язык.*

Доказательство. Надо добавить 2 леммы и док-во. □

Билет 15.

Синтаксический анализ за кубическое время: алгоритм Кокка–Касами–Янгера, построение дерева разбора.

Лекция 7. Грамматики над односимвольным алфавитом. Невыразимые действия над грамматиками. Удаление пустых правил, удаление единичных правил, нормальный вид Хомского. Синтаксический анализ за кубическое время, построение дерева разбора.

Конспект, запись.

Теорема 19. *Для всякой грамматики $G = (\Sigma, N, R, S)$ есть алгоритм синтаксического анализа, работающий за время $O(n^3)$, где n - длина входной строки.*

Сами алгоритмы и рассуждения о них - в третьем параграфе конспекта.

Билет 16.

Синтаксический анализ, использующий память $O((\log n)^2)$. Его параллельная реализация схемой.

Лекция 8. Синтаксический анализ за кубическое время, построение дерева разбора. Параллельный разбор за время (log n)2. Неразрешимые задачи для грамматик

Конспект, запись.

Билет 17.

Неразрешимые задачи для грамматик.

Лекция 8. Синтаксический анализ за кубическое время, построение дерева разбора. Параллельный разбор за время (log n)2. Неразрешимые задачи для грамматик

Конспект, запись.

Впадлу мне чёто писать эти билеты. Один - первая половина конспекта, второй - вторая.

Гиршовская часть.

Эх щас бы пивка холодного ...

Билет 1.

Универсальная машина Тьюринга, эффективное моделирование k-ленточной ДМТ на двухленточной ДМТ.

Лекция 1. Введение в теорию сложности вычислений.

Слайды, запись.

Определение 15. *Универсальная машина Тьюринга* $M(a, x)$ берёт описание (номер) a машины M_a и её вход x, и выдаёт тот же результат, что и $M_a(x)$.

Теорема 20. *k-ленточную машину, работающую $T(n)$ шагов можно промоделировать за $O(T(n)\log T(n))$ шагов.*

Доказательство. Сначала покажем, что моделирование машины можно свести к моделированию стека. Каждая лента машины Тьюринга разбивается на два стека: символы слева от головки и символы справа от головки. Каждая операция с лентой соответствует нескольким операциям со стеками. k лент представляются в виде $2k$ стеков, все стеки мы будем хранить на одной ленте, к примеру, можно разделить клетки ленты между стеками через $2k$ (альтернативно можно укрупнить алфавит так, чтобы в одном символе умещалось $2k$ символов). Мы покажем, что мы сможем выполнить $T(n)$ операций с одним стеком за $O(T(n)\log T(n))$ шагов, причем после каждой операции вторая лента будет пустая и обе головки будут расположены в начале ленты. Тогда на моделирование работы $2k$ стеков уйдет тоже $O(T(n)\log T(n))$ шагов, так как k — константа.

Прежде, чем описывать, как устроен стек, решим другую задачу. Предположим, мы хотим хранить счетчик с операциями ± 1. Причем хочется, чтобы среднее число замен цифр счетчика на одну операцию было бы константой. Если счетчик хранить просто в бинарном виде, то если N раз подряд с числом вида $2n-1$ совершать пару действий $+1, -1$, то число меняющихся разрядов будет не меньше Nn. Поэтому будем использовать ленивую систему, в которой цифры будут 0, 1 и 2. Представление в такой системе счисления будет неоднозначным, к примеру $1 \cdot 2^1 = 2 \cdot 2^0$. Пусть у нас есть число $K = \sum_{i=0}^{n} a_i 2^i$, где $a_i \in \{0, 1, 2\}$. Покажем, как к нему прибавлять единицу. Мы находим наименьшее такое k, что $a_k < 2$, число $K + 1$ будет представлено в виде $\sum_{i=0}^{k-1} 2^i + (a_k + 1) \cdot 2^k + \sum_{k<i\leq n} a_i 2^i$. Аналогично, чтобы отнять единицу, мы найдем наименьшее такое k, что $a_k > 0$, число $K - 1$ будет представлено в виде $\sum_{i=0}^{k-1} 2^i + (a_k - 1) \cdot 2^k + \sum_{k<i\leq n} a_i 2^i$. Заметим, что если операция прибавления или вычитания единицы затронула k-й разряд, то в следующий раз k-й разряд будет затронут как минимум через 2^k операций, так как после операций, в которой участвовал k-й разряд все предыдущие разряды равны единице. Итого, если мы делали T операций ± 1, то k-й разряд затрагивался не больше, чем $\lceil \frac{T}{2^k} \rceil$ раз. Значит, всего затронутых разрядов не больше $\sum_{k\geq 0} \lceil \frac{T}{2^k} \rceil < 2T \sum_{k\geq 0} \frac{1}{2^k} = 4T$, т.е. не больше, чем 4 разряда на одну операцию.

Теперь опишем, как мы будем хранить стек. Стек мы разобьем на блоки с номерами $0,1,2,\ldots$. Блок с номером k состоит из двух зон размера 2^k. Каждый блок может быть пуст, либо полностью заполнены обе зоны, либо заполнен наполовину, это соответствует цифрам 0, 1 и 2 соответственно. Содержимое стеков получится, если выписать подряд содержимое блоков с нулевого, причем сначала выписывается первая зона, затем вторая. Мы будем поддерживать инвариант, что наполненность блоков (0,1,2) задает число в ленивой двоичной системе счисления. Чтобы добавить элемент в стек, нужно найти блок с минимальным номером, который заполнен не полностью, если требуется, то сдвинуть элементы так, чтобы свободной была первая зона, поместить в первую зону половину элементов из предыдущих блоков (сохраняя порядок), остальные элементы раскидать так, чтобы каждый блок был заполнен ровно наполовину. Аналогично можно определить операцию удаления элемента из стека.

Операции добавления и удаления элемента реализуются с помощью второй ленты, на которую копируются все элементы из блоков, которые просматриваются. Операции удаления и добавления элементов, которые затрагивают k блоков можно реализовать за $O(2^k)$ операций. Между двумя операциями, которые затрагивают k-й блок прохо дит не меньше 2^k операций. Таким образом, чтобы выполнить T операций со стеком, нам понадобится $C\sum_{k=0}^{l} 2^k \frac{T}{2^k}$ операций, где $l = O(\log T(n))$ — число блоков, которые можно успеть затронуть за T шагов. Таким образом общее число операций оценивается $O(T(n)\log T(n))$. $\square$

Билет 2.

Теоремы об иерархии по времени и памяти для детерминированных вычислений.

Лекция 1. Введение в теорию сложности вычислений.

Слайды, запись.

Определение 16. Функция f называется *конструируемой по времени*, если существует детерминированная машина Тьюринга, вычисляющая (двоичное представление) $f(n)$ и затрачивающее на это память не более $f(n)$.

Теорема 21. *$DSpace(S_1(n)) \neq DSpace(S_2(n))$, где $S_1(n) = o(S_2(n))$ и $S_1(n) \geq \log n$ для всех $n > n_0$.*

Доказательство. Рассмотрим следующий язык (обозначая $\langle M \rangle$ запись машина M - т.е., по существу, её функцию перехода):

$$x = \langle M \rangle 01^k \left(\begin{array}{c} k \in \mathbb{N} \cup \{0\}, \\ |\langle M \rangle| < \frac{1}{10} S_2(|x|), \\ M \text{ отвергает } x \text{ с памятью } \leq \frac{1}{10} S_2(|x|) \end{array} \right)$$

Этот язык мы можем распознать, используя $S_2(|x|)$ памяти. Покажем, что его нельзя распознать, потратив только $S_1(|x|)$ памяти. Действительно, пусть существует машина M_1, которая производит такое распознавание. Тогда для некоторого N и для всех $n > N$ справедливо $S_1(n) < \frac{1}{10} S_2(n)$. Рассмотрим число n, болшее N и большее $|\langle M \rangle|$. Если строка M_1 принимает строку $\langle M \rangle 01^{n - |\langle M \rangle| - 1}$, то она по определению нашего языка ему не принадлежит, а если отвергает, то принадлеэит! □

Теорема 22. *$DTime(t(n)) \neq DTime(T(n))$, где $t(n) \log t(n) = o(T(n))$, $T(n) = \Omega(n)$.*

Доказательство.

$$x = \langle M \rangle 01^k \left(\begin{array}{c} k \in \mathbb{N} \cup \{0\}, \\ |\langle M \rangle| < T(|x|), \\ M \text{ отвергает } x \text{ с памятью } \leq \frac{T(|x|)}{\log T(|x|)} \cdot \log \frac{T}{\log T} \end{array} \right)$$

Такой язык лежит в $DTime(|x|)$. Здесь используется эффективная универсальная МТ, моделирующая $f(n)$ шагов произвольной машины (с произвольным числом лент) за $O(f(n) \log f(n))$ шагов. Остальное - аналогично иерархии по памяти. □

Предметный указатель