Теоретическая информатика.

Лектор — А.С. Охотин, Э.А. Гирш и Д.О. Соколов
Создатель конспекта — Глеб Минаев *

TODOs

Содержание

Литература:

- ...

Страницы курса:

- Часть курса, прочитанная А.С. Охотиным.

1 Вычислимость

Определение 1. *Машина Тьюринга* — это реализация понятия вычисления. Она заключается в том, что имеется бесконечная в обе стороны клетчатая лента с записанным на ней конечным словом (по букве на клетку) — входные данные вычисления — и головка — вычисляющий аппарат, которая в каждый момент времени смотрит на какую-то клетку ленты и имеет в себе некоторое внутреннее состояние вычислений. Каждым своим действием она читает символ в клетке, на которую смотрит, и в зависимости от прочитанного символа и внутреннего состояния в данный момент она пишет в клетку, на которую смотрит новый символ, стирая старый, переходит на новую клетку и меняет внутренне состояние.

Формально машина Тьюринга M — это совокупность

- входного алфавита Σ — (конечного) алфавита входных данных,
- рабочего алфавита Γ — (конечного) алфавита, над которым мы работаем, что $\Gamma \supseteq \Sigma \sqcup \{\_\}$,

*Оригинал конспекта расположен на GitHub. Также на GitHub доступен репозиторий с другими конспектами.

- конечного множества (внутренних) состояний Q,
- начального состояния $q_0 \in Q$,
- функции переходов $\delta : Q \times \Gamma \to Q \times \Gamma \times \{-1; +1\}$
- и дополнительных объектов и условий в зависимости от предназначения машины.

Входными данными программы является конечная строка $s_1 \dots s_l$, которая превращается в запись $\{a_i\}_{i \in \mathbb{Z}}$ на ленте по правилу

$$a_i = \begin{cases} s_{i+1} & \text{если } 0 \leqslant i \leqslant l-1, \\ \text{␣} & \text{иначе.} \end{cases}$$

Состояние каждого момент вычисления — это совокупность

- состояния ленты $\{a_i\}_{i \in \mathbb{Z}}$, где $a_i \in \Sigma$ и все a_i кроме конечного числа элементов являются пробелом ␣,
- положения головки $n \in \mathbb{Z}$ на ленте и
- состояния головки $q \in Q$.

Состояние начального момента вычислений образуется из

- состояния ленты, полученного из входных данных,
- положения головки по умолчанию $n = 0$
- начального состояния головки q_0.

Каждым шагом состояние $(\{a_i\}_{i \in \mathbb{Z}}, n, q)$ заменяется на состояние $(\{a'_i\}_{i \in \mathbb{Z}}, n', q')$, где $a'_i = a_i$ для всех $i \in \mathbb{Z} \setminus \{n\}$ и

$$(q', a'_n, n' - n) := \delta(q, a_n).$$

Далее есть два вида предназначений машины:

1. **Распознование свойства.** В этом случае мы хотим уметь распознавать разные конечные строки, т.е. чтобы машина отвечала “да” или “нет” на вопрос “Лежит ли эта строка в заданном семействе?”. В таком случае у машины выделяются *принимающее состояние* $q_{acc} \in Q$ и *отвергающее состояние* $q_{rej} \in Q$, и когда машина переходит в одно из этих двух состояний, вычисления прекращаются. В таком случае $\delta(q, a)$ должно быть не определено в случае $q \in \{q_{acc}; q_{rej}\}$.
2. **Вычисление функции.** В этом случае мы хотим вычислять значение некоторой функции $M : \Sigma^* \to \Sigma^*$, т.е. чтобы машаина после некоторого количества действий говорила “Готово.” и оставляла на ленте конечное слово в том же формате, в котором производится ввод. В таком случае у машины выделяется *состояние останова* $q_{halt} \in Q$, и когда машина переходит в это состояние, вычисления прекращаются, а на ленте должно остаться слово в правильном формате. В таком случае $\delta(q, a)$ должно быть не определено в случае $q = q_{halt}$.

Замечание. Фактически множество результатов работы машины состоит из переходов в одно из терминальных состояний (q_{acc}, q_{rej} или q_{halt}) и попадание в “вечный цикл”.

Теорема 1. *В определении машина Тьюринга множество возможных смещений головки* $\{+1; -1\}$ *можно сменить на некоторое* $S \subseteq \mathbb{Z}$.

1. *Тогда если* S *конечно и всякое целое число представляется в виде суммы хотя бы одного значения (возможно, с повторами) из* S*, то определение получается равносильным.*

2. *То же самое верно для всякого* S *(не обязательно конечного).*

Доказательство.

1. Действительно, пусть у нас имеется машина с множеством возможных сдвигов S_1, а у нас есть множество возможных сдвигов S_2, что всякое число из S_1 представляется в виде суммы некоторых (хотя бы одного и, возможно, с повторениями) членов из S_2. Тогда для всякой перехода между состояниями $q_1 \to q_2$ можно создать такие фальшсостояния, что смещение на значение из S_1 будет заменено на несколько смещений на значения из S_2 с тем же итогом. Таким образом мы можем всякую машину на S_1 поменять на машину на S_2 и большим числом состояний.

   Таким образом множества смещений S_1 и S_2 реализуют равносильные модели, если всякое значение из S_1 расскладывается в сумму значений S_2 и наоборот. При для $S_1 = \{+1; -1\}$ переход $S_2 \to S_1$ очевиден, а переход $S_1 \to S_2$ означает, что 1 и -1 раскладываются в суммы некоторых чисел из S_2, что равносильно разложимости всякого целого.

2. Несложно понять, что в каждой конкретной машине Тьюринга множество $Q \times \Sigma$ конечно, а значит S можно заменить на конечное подмножества, что задача для данной машины не изменится.

□

Следствие 1.1. *Вместо* $\{+1; -1\}$ *для удобства можно также подразумевать* $\{+1; 0; -1\}$.

Замечание. В данный момент мы будем рассматривать только задачу распознавания свойства.

Определение 2. *Язык машины* M или *язык, распознаваемый машиной* M, — это множество $L(M) \subseteq \Sigma^*$ входных строк, которые машина принимает. Т.е. это множество всех конечных строк $w \in \Sigma^*$ таких, что машина M на входе w завершает работу и принимает данный вход.

Язык, распознаваемый какой-нибудь машиной, называется *рекурсивно-перечислимым.*

Язык, распознаваемый какой-нибудь машиной, которая останавливается на любом входе, называется *рекурсивным.*

Пример 1. Пусть $\Sigma = \{a; b\}$, $L = \{a^n b^n\}_{n \geqslant 0}$. Тогда L распознаётся машиной Тьюринга (которая причём останавливается на всяком входе!).

Действительно, давайте напишем машину, которая будить ездить из одного конца нашей строки в другую и стирать a справа и b слева, и, если получит пустую строку, примет вход, а если поймет, что в какой-то момент получается что-то неправильное, то отвергнет вход. Формально, возьмём $Q = \{q_0; q_1; q_2; q_3; q_{acc}; q_{rej}\}$ и $\Gamma = \{a; b; \text{␣}\}$, а функцию перехода опишем следующей таблицей:

$Q \backslash \Gamma$	a	b	␣
q_0	q_1, ␣, $+1$	q_{rej}	q_{acc}
q_1	q_1, a, $+1$	q_1, b, $+1$	q_2, ␣, -1
q_2	q_{rej}	q_3, ␣, -1	q_{rej}
q_3	q_3, a, -1	q_3, b, -1	q_0, ␣, $+1$

Т.е. во с помощью q_0 и q_2 мы стираем a слева и b справа соответственно, а с помощью q_1 и q_2 перемещаемя до конца вправо и влево соответственно.

Теорема 2. *Есть язык, нераспознаваемый никакой машиной Тьюринга.*

Доказательство. Несложно видеть, что для всякого конечного непустого алфавита Σ количество $|\Sigma^*|$ конечных строк над Σ счётно, а значит языков над Σ континуум. При этом машин Тьюринга с входным алфавитом Σ счётное число. Значит почти все языки не распознаются машинами Тьюринга.

При этом есть довольно простые конкретные примеры нераспознаваемых языков. □

Определение 3. *Запись $\sigma(M)$ машины Тьюринга M* — это конечной битовая строка, созданная по следующему правилу. Пусть у машины $M = (\Sigma, \Gamma, Q, q_0, \delta, q_{acc}, q_{rej})$

- $\Sigma = \{a_1; \ldots; a_l\}$,
- $\Gamma = \Sigma \cup \{a_{l+1}; \ldots; a_m\}$, где причём $a_m = \text{␣}$,
- $Q = \{q_0; \ldots; q_{n-1}\}$, где причём $q_{acc} = q_{n-2}$, $q_{rej} = q_{n-1}$.

Тогда запишем всё в унарной системе счисления, т.е. (всякий абстрактный) массив числовых данных будет храниться в виде последовательностей единиц, длины которых будут равняться соответствующим значениям массиве, разделённых нулями. Тогда машина M будет записана строкой

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

(где умножение, безусловно, — конкатенация, а $\prod$ — конкатенация нескольких строк).

Определение 4. Языки L_0 и L_1 — это

$$L_0 := \{\sigma(M) \mid \sigma(M) \notin L(M)\} \qquad \text{и} \qquad L_1 := \{\sigma(M) \mid \sigma(M) \in L(M)\}.$$

Теорема 3.

1. *L_0 не рекурсивно-перечислимый.*
2. *L_1 рекурсивно-перечислимый.*
3. *L_1 не рекурсивный.*

Доказательство.

1. Предположим противное, т.е. есть машина M, распознающая L_0. Тогда если M принимает $\sigma(M)$, то $\sigma(M) \in L(M)$, но тогда $\sigma(M) \notin L$. Если же M не принимает $\sigma(M)$, то $\sigma(M) \notin L(M)$, но тогда $\sigma(M) \in L$. Противоречие наподобие парадокса Рассела. Значит нет никакой машины, распознающей L_0.

2. Неформально опишем машину, которая будет распознавать L_1. Идея машины заключается в том, что получая на вход $\sigma(M)$, после этого запсанного кода машины M она запишет код начального состояния q_0 машины M, а затем код $\sigma(M)$ как входную строчку для M. После этого она начнёт моделировать работу машины M на входе $\sigma(M)$. Состояние головки будет писаться перед клеткой, на которую головка смотрит. Если места слева будет не хватать, то вся запись будет просто сдвигаться вправо.

3. Покажим, что всякая машина, распознающая L_1 зацикливается при некотором входе. Предположим противное, т.е. есть машина $\widetilde{M}$, которая распознаёт L_1 и всегда останавливается. Тогда построим $\widehat{M}$ по алгоритму:

- Проверить правда ли, что на было подано некоторое $\sigma(M)$. Если нет, то отвергнуть.
- Работать как $\widetilde{M}$ на $\sigma(M)$. Если $\widetilde{M}$ принимает, то отвергнуть. Если отвергает — принять.

Тогда $\widehat{M}$ распознаёт L_0 — противоречие. Значит машины $\widetilde{M}$ не существует.

□

Теорема 4. *Язык*

$$L_\varnothing := \{\sigma(M) \mid L(M) = \varnothing\}$$

не рекурсивно-перечислимый.

Доказательство. Предположим противное, т.е. есть машина $M_\varnothing$, которая распознаёт $L_\varnothing$. Тогда построим машину M_0 по следующему алгоритму.

1. Проверить правда ли, что на было подано некоторое $\sigma(M)$. Если нет, то отвергнуть.
2. Построить $\sigma(M')$, где машина M' работает по следующему алгоритму.
   - Стереть входную строку.
   - Написать $\sigma(M)$.
   - Запустить M.
3. Запустить $M_\varnothing$ на $\sigma(M')$.

M' принимает любую строку, если M принимает $\sigma(M)$, отвергает любую строку, если M отвергает $\sigma(M)$, и зацикливается, если M зацикливается на $\sigma(M)$. Следовательно $M_\varnothing$ примет $\sigma(M')$ тогда и только тогда, когда $\sigma(M) \notin L(M)$. Таким образом M_0 распознаёт L_0. □

Определение 5. Для всякого свойства P языков можно определить язык

$$L_P := \{\sigma(M) \mid L(M) \text{ обладает свойством } P\}.$$

Можно считать, что свойство есть некоторое множество языков (т.е. подмножество 2^{Σ^*}), а обладание свойством означает содержание в нём как в множестве.

Свойство называется *тривиальным*, если либо ни один рекурсивно-перечислимый язык, либо все рекурсивно-перечислимые языки обладают этим свойством.

Пример 2.

1. Если P — это “непустота”, то

   $$L_P := \{\sigma(M) \mid L(M) \neq \varnothing\}.$$

   Такой язык рекурсивно-перечислим.

2. $$L_P := \{\sigma(M) \mid M \text{ отвергает конечное число программ на C++}\}.$$

3. $$L_P := \{\sigma(M) \mid L(M) = L_0\}.$$

   Как мы уже знаем $L_P = \varnothing$, т.е. данное свойство тривиально.

Теорема 5 (Райса)**.** *Всякое нетривиальное свойство нерекурсивно.*

Доказательство. Пусть дано нетривиальное свойство $P \subseteq 2^{\Sigma^*}$ рекурсивно-перечислимых языков. Предположим противное: есть машина M_P распознаёт L_p (не зацикливается и принимает только язык L_p).

Предположим, что $\varnothing \notin P$. Тогда есть машина $\widehat{M}$, что $L(\widehat{M}) \notin P$. Тогда построим машину Тьюринга M_1, принимающую в себя $\sigma(M)$ некоторой машины Тьюринга M, со следующим алгоритмом:

1. Построить машину Тьюринга $\widetilde{M}$ по следующему описанию.
   (a) Сохранить вход w.
   (b) Запустить M на $\sigma(M)$.
   (c) Если отвергнет, зациклиться. Если примет, запустить $\widehat{M}$ на w.
2. Запустить M_P на $\sigma(\widetilde{M})$.

Заметим, что если $\sigma(M) \notin L(M)$, то машина $\widetilde{M}$ зацикливается на любом входе, т.е. $L(\widetilde{M}) = \varnothing$. Иначе $\widetilde{M}$ совпадает с $\widehat{M}$, т.е. $L(\widetilde{M}) = L(\widehat{M})$. Таким образом

$$L(\widetilde{M}) \in P \quad \Longleftrightarrow \quad \sigma(M) \in L(M).$$

Таким образом ответ M_p на входе $\sigma(\widetilde{M})$ есть ответ на вопрос принадлежности $\sigma(M)$ множеству $L(M)$. Таким образом M_1 не зацикливается и распознаёт язык L_1, что противоречит ранее доказанным утверждениям.

Теперь предположим $\varnothing \in P$. Тогда есть машина $\widehat{M}$, что $L(\widehat{M}) \notin P$. По аналогии можно построить машину, которая распознаёт L_0. □

2 Формальные зыки и автоматы

Определение 6. *Алфавит* Σ — фиксированный (контекстом) набор элементов (*символов*). Мы будем рассматривать только конечные алфавиты.

Строка — (если не оговорено обратное, конечная) последовательность символов из алфавита. Строки обозначаются либо просто как переменные, т.е. w, либо в виде

$$a_1 \dots a_n,$$

где a_i — символы из алфавита Σ. Строка, где $n = 0$, называется *пустой строкой* и обозначается ε. Значение n называется длиной строки w и обозначается $|w|$. Количество вхождение некоторого символа a (т.е. количество индексов i, что $a_i = a$) обозначается $|w|_a$. Множество всех строк обозначается Σ^*.

Конкатенация — бинарная операция на строках, определяемая по правилу

$$(a_1 \dots a_n, b_1 \dots b_m) \mapsto a_1 \dots a_n b_1 \dots b_m.$$

Конкатенация строк u и v обозначается uv или $u{\cdot}v$. Множество строк с операцией конкатенации и выделенной ε образуют моноид.

Обращение строки — унарная операция на строках, определяемая по правилу

$$a_1 \dots a_n \mapsto a_n \dots a_1.$$

Обращение строки w обозначается w^R.

Язык — множество строк, т.е. подмножество Σ^*.

Определение 7. *Детерминированный конечный автомат* (также “ДКА” или “DFA”) A — это совокупность

- входного алфавита Σ,
- конечного множества состояний Q,
- начального состояния $q_0 \in Q$,
- множество принимающих состояний $F \subseteq Q$,
- функция перехода $\delta : Q \times \Sigma \to Q$.

Входными данными является конечная строка $w = s_1 \dots s_n$. Состояние вычисления после k шагов — это некоторое состояние автомата $q_k \in Q$. Соответственно, начальное состояние вычисления (оно же состояние вычисления после 0 шагов) — начальное состояние автомата q_0. Каждым шагом состояние q_k заменяется на $q_{k+1} := \delta(q_k, s_{k+1})$. Т.е. формально вычисление — это последовательность состояний $(q_k)_{k=0}^n$, где

- q_0 — начальное состояние автомата,
- $q_{k+1} := \delta(q_k, s_{k+1})$.

Состояние q_n называется *результатом вычисления*. Также говорят, что автомат принимает строку w, если $q_n \in F$, и отвергает в ином случае.

Язык автомата A или *язык, распознаваемый автоматом* A, — это множество $L(A)$ всех строк, распознаваемых (принимаемых) автоматом A. Язык, распознаваемый хоть каким-нибудь автоматом называется *регулярным*.

Также в качестве удобства обозначений определим функцию δ^* на $Q \times \Sigma^*$ как

$$\delta^*(q, a_1 \dots a_n) := \delta^*(\dots \delta(q, a_1), \dots, a_n),$$

т.е. δ^* задаётся рекурсивно как

$$\delta^*(q, \varepsilon) = q, \qquad \delta^*(q, aw) = \delta^*(\delta(q, a), w).$$

Иногда мы будем писать δ, подразумевая δ^*.

Замечание. Детерминированные конечные автоматы удобно изображать в виде ориентированного графа. Пусть Q — вершины графа, и из каждой вершины q ведёт по $|\Sigma|$ рёбер: ребро из q с меткой s (для всякого $s \in \Sigma$) ведёт в $\delta(q, s)$. Также небольшой стрелочкой ведущей из ниоткуда в q_0 удобно помечать q_0 как начальную вершину, и удобно обвести каждую вершину из F, чтобы пометить её как принимающую.

Замечание. Несложно видеть, что

$$L(A) = \{w \in \Sigma^* \mid \delta_A^*(q_{A,0}, w) \in F_A\}.$$

Это также значит, что $w \in L(A)$ тогда и только тогда, когда при проходе в графе A по пути, соответствующему слову w, мы попадаем в принимающее состояние. Говоря иначе, есть некоторая последовательность состояний $q_0, \dots, q_n$, что

- $n = |w|$,
- $q_0 = q_{A,0}$ — начальное состояние автомата,

- $q_{k+1} = \delta_A(q_k, a_{k+1})$,
- $q_n \in F_A$.

Теорема 6. *Язык $L := \{a^n b^n\}_{n\in\mathbb{N}}$ нерегулярен.*

Доказательство. Предположим имеется ДКА A, что $L(A) = L$. Заметим, что тогда строка $a^{|Q|}b^{|Q|} \in L$. Пусть вычисление этой строки имеет вид $q_0 \dots q_{|Q|} \dots$ (понятно, что $q_{|Q|}$ — состояние вычисления после последней буквы a). По принципу Дирихле есть из состояний q_0, …, $q_{|Q|}$ есть два совпадающих; пусть это будут q_i и q_j $(i < j)$. Тогда покажем, что $a^{|Q|-(j-i)}b^{|Q|}$ тоже распознаётся (хотя не должна). Заметим, что процесс вычисления букв a будет иметь вид

$$q_0 \dots q_i = q_j \dots q_{|Q|}.$$

Таким образом состояние после прочтения последней буквы a будет тем же, а значит состояние после прочтения всей строки тоже будет тем же, а значит принимающим. □

Определение 8. Недетерминированный конечный автомат (также "НКА" или "NFA") — это совокупность

- входного алфавита Σ,
- конечного множества состояний Q,
- множества начальных состояния $S \subseteq Q$,
- множество принимающих состояний $F \subseteq Q$,
- функция перехода $\delta : Q \times \Sigma \to 2^Q$.

Входными данными является конечная строка $w = s_1 \dots s_n$. Состояние вычисления после k шагов — это некоторое множество состояний автомата $S_k \in Q$. Соответственно, начальное состояние вычисления (оно же состояние вычисления после 0 шагов) — множество начальных состояние автомата $S_0 = S$. Каждым шагом состояние S_k заменяется на $S_{k+1} := \bigcup_{q\in S_k} \delta(q, s_{k+1})$. Т.е. формально вычисление — это последовательность множеств состояний $(S_k)_{k=0}^n$, где

- $S_0 = S$ — начальное состояние автомата,
- $S_{k+1} := \bigcup_{q\in S_k} \delta(q, s_{k+1})$.

Множество состояний S_n называется *результатом вычисления*. Также говорят, что автомат принимает строку w, если $S_n \cap F \neq \varnothing$, и отвергает в ином случае.

Язык автомата A или *язык, распознаваемый автоматом A*, — это множество $L(A)$ всех строк, распознаваемых (принимаемых) автоматом A. Язык, распознаваемый хоть каким-нибудь автоматом называется *регулярным*.

Также в качестве удобства обозначений определим функцию δ^* на $2^Q \times \Sigma^*$ рекурсивно как

$$\delta^*(T, w) = \bigcup_{q\in T} \delta^*(q, w), \qquad \delta^*(\{q\}, \varepsilon) = \{q\}, \qquad \delta^*(\{q\}, aw) = \delta^*(\delta(q, a), w).$$

Иногда мы будем писать δ, подразумевая δ^*.

Замечание. Недетерминированные конечные автоматы удобно изображать в виде ориентированного графа. Пусть Q — вершины графа, и из каждой вершины q ведёт некоторое количество рёбер: по ребру из q с меткой s (для всякого $s \in \Sigma$) ведёт в каждую вершину из $\delta(q, s)$. Также небольшими стрелочками ведущими из ниоткуда в каждую вершину из S удобно помечать S как множество начальных вершин, и удобно обвести каждую вершину из F, чтобы пометить её как принимающую.

Замечание. Несложно видеть, что $w \in L(A)$ тогда и только тогда, когда есть путь в графе A, соответствующий строке w, что конечное состояние является принимающим. Говоря иначе, есть некоторая последовательность состояний $q_0, \ldots, q_n$, что

- $n = |w|$,
- $q_0 \in S_A$,
- $q_{k+1} \in \delta_A(q_k, a_{k+1})$,
- $q_n \in F_A$.

С другой стороны это также равносильно тому, что $\delta_A^*(S_A, w) \in F_A$.

Теорема 7. *У всякого недетерминированного автомата A есть детерминированный автомат A', что $L(A') = L(A)$.*

Доказательство. Пусть дано, что $A = (\Sigma, Q, S, \delta, F)$. Тогда определим

$$A' = (\Sigma, 2^Q, S, \delta', \{T \subseteq Q \mid T \cap F \neq \varnothing\}),$$

где δ' определяется по правилу

$$\delta'(T, s) := \delta(T, s) = \bigcap_{q \in T} \delta(q, s).$$

Проще говоря, A' имитирует A, запоминая в каком множестве состояний мы находимся в каждый момент и эмулируя правильный переход к новому множеству состояний. Формально говоря, несложно показать по индукции по $|w|$, что множество состояний автомата A после прочтения слова w является состоянием автомата A' после прочтения того же слова w. □

Замечание 1. Можно на языках ввести следующие операции.

- $K \cup L$, $K \cap L$, $K \triangle L$, $\overline{L}$ — стандартные операции на множествах.
- Конкатенация. $KL = K \cdot L := \{uv \mid u \in K \wedge v \in L\}$.
- Конкатенация нескольких копий. $L^k := L \cdot \cdots \cdot L$. $L^0 = \{\varepsilon\}$.
- Конкатенация любого конечного числа копий или звёздочка Клини. $L^* = \bigcup_{k=0}^{\infty} L^k$.

В таком случае $(2^{\Sigma^*}, \cup, \cdot, \varnothing, \{\varepsilon\})$ — полукольцо.

2.1 Регулярные выражение и равносильность конечным автоматам

Определение 9. Рекурсивно определим понятие *регулярного выражения.*

- $\varnothing$ — регулярное выражение.
- a, где $a \in \Sigma$, — регулярное выражение.
- $(\varphi\psi)$, $(\varphi|\psi)$ и φ^*, где φ и ψ есть регулярные выражения, — регулярные выражения.

Множество регулярных выражений над алфавитом Σ обозначается $\mathrm{RE}(\Sigma)$.

Теперь рекурсивно определим *язык регулярного выражения.*

- $L(\varnothing) = \varnothing$.

- $L(a) = \{a\}$.
- $L(\varphi\psi) = L(\varphi)L(\psi)$.
- $L(\varphi|\psi) = L(\varphi) \cup L(\psi)$.
- $L(\varphi^*) = L(\varphi)^*$.

Теорема 8. *Язык распознаётся конечным автоматом тогда и только тогда, когда задаётся регулярным выражением.*

Определение 10. Недетерминированный конечный автомат с ε-переходами (также "ε-НКА" или "ε-NFA") — это совокупность

- входного алфавита Σ,
- конечного множества состояний Q,
- множества начальных состояния $S \subseteq Q$,
- множество принимающих состояний $F \subseteq Q$,
- функция перехода $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$.

ε-замыканием состояния q называется множество состояний

$$E(q) := \{p \mid \exists k \in \mathbb{N}\colon \exists q_0, \ldots, q_k \in Q\colon q_0 = q \wedge q_k = p \wedge \forall i = 1, \ldots, k\ q_i = \delta(q_{i-1}, \varepsilon)\}.$$

Также по аналогии ε-замыканием множества состояния T называется множество состояний

$$E(T) := \bigcup_{q \in T} E(q),$$

откуда, в частности, получается "рекурсивное определение"

$$E(T) := T \cup \bigcup_{q \in T} E(\delta(q, \varepsilon)).$$

Входными данными является конечная строка $w = s_1 \ldots s_n$. Состояние вычисления после k шагов — это некоторое множество состояний автомата $S_k \in Q$. Соответственно, начальное состояние вычисления (оно же состояние вычисления после 0 шагов) — множество начальных состояние автомата $S_0 = E(S)$. Каждым шагом состояние S_k заменяется на $S_{k+1} := \bigcup_{q \in S_k} E(\delta(q, s_{k+1}))$. Т.е. формально вычисление — это последовательность множеств состояний $(S_k)_{k=0}^n$, где

- $S_0 = E(S)$ — начальное состояние автомата,
- $S_{k+1} := \bigcup_{q \in S_k} E(\delta(q, s_{k+1}))$.

Множество состояний S_n называется *результатом вычисления*. Также говорят, что автомат принимает строку w, если $S_n \cap F \neq \varnothing$, и отвергает в ином случае.

Язык автомата A или *язык, распознаваемый автоматом A*, — это множество $L(A)$ всех строк, распознаваемых (принимаемых) автоматом A. Язык, распознаваемый хоть каким-нибудь автоматом называется *регулярным*.

Также в качестве удобства обозначений определим функцию δ^* на $2^Q \times \Sigma^*$ рекурсивно как

$$\delta^*(T, w) = \bigcup_{q \in T} \delta^*(q, w), \qquad \delta^*(\{q\}, \varepsilon) = E(\{q\}), \qquad \delta^*(\{q\}, aw) = \delta^*\left(\bigcup_{p \in E(q)} \delta(p, a), w\right).$$

Иногда мы будем писать δ, подразумевая δ^*.

Замечание. Недетерминированные конечные автоматы с ε-переходами удобно изображать точно также как обычные НКА, но на стрелках (ориентированных рёбрах) переходов теперь можно писать и новодобавленный символ ε.

Замечание. Несложно видеть, что $w \in L(A)$ тогда и только тогда, когда есть некоторое представление $w = s_1 \dots s_n$, где $s_i \in \Sigma \cup \{\varepsilon\}$, и некоторая последовательность состояний $q_0, \dots, q_n$, что

- $q_0 \in S_A$,
- $q_{k+1} \in \delta_A(q_k, s_{k+1})$,
- $q_n \in F_A$.

С другой стороны это также равносильно тому, что $\delta_A^*(S_A, w) \in F_A$.

Лемма 9. *Всякое регулярное выражение можно заменить на ε-НКА, порождающий тот же язык.*

Доказательство. Давайте построим для каждого регулярного выражения ε-НКА с ровно одним начальным состоянием и ровно одним принимающим состоянием. И будем мы это делать по рекурсивному построению регулярного выражения:

- Автомат для регулярного выражения $\varnothing$ будет состоять только из одного начального и одного принимающего состояний, не иметь других состояний и не иметь никаких переходов.
- Автомат для регулярного выражения a ($a \in \Sigma$) будет состоять только из одного начального и одного принимающего состояний, не иметь других состояний и иметь единственный переход от начального состояния к принимающему по символу a.
- Автомат для регулярного выражения $\varphi\psi$ будет получаться проведением перехода по ε от принимающего состояния φ к начальному состоянию ψ. Соединённые состояния теряют свои роли (т.е. больше не являются принимающей и начальной соответственно).
- Автомат для регулярного выражения $\varphi|\psi$ будет получаться проведением переходов по ε от начального состояния строимого автомата к начальным состояниям автоматов φ и ψ и переходов по ε от принимающих состояний автоматов φ и ψ к принимающему состоянию строимого автомата. Начальные и принимающие состояния старых автоматов теряют свои роли.
- Автомат для регулярного выражения $\varphi*$ будет получаться проведением перехода по ε от начального состояния строимого автомата к начальному состоянию автомата φ, перехода по ε от принимающего состояния автомата φ к начальному состоянию строимого автомата и перехода по ε от начального состояния строимого автомата к принимающему состоянию строимого автомата. Начальные и принимающие состояния старого автомата теряют свои роли.

Картиночки.

Несложно видеть по индукции по построению выражения, что строимые автоматы, действительно, строят язык соответствующих регулярных выражений. □

Лемма 10. *Всякий ε-НКА можно заменить на НКА, порождающий тот же язык.*

Доказательство. Неформально говоря, мы хотим во всяком пути $s_1 \dots s_n$ произвести разбиение на несколько блоков вида $\varepsilon^k a$ (для некоторых $k \in \mathbb{N}$ и $a \in \Sigma$) оканчивающееся на дополнительный блок вида ε^k и заменить каждый (не дополнительный) блок на ровно один переход; дополнительный блок можно будет убрать при правильной замене множества принимающих состояний.

Теперь формальное построение. Пусть дан ε-НКА $A = (\Sigma, Q, S, F, \delta)$. Будем строить НКА A'. Σ, Q и S оставим без изменений. Теперь определим новые δ' и F'. δ' определяется по правилу

$$\delta'(q, a) := \bigcup_{p \in E(q)} \delta(p, a),$$

а F' определяется как

$$F' := \{q \in Q \mid E(q) \cap F \neq \varnothing\}.$$

□

Определение 11. Недетерминированный конечный автомат с regex-переходами (также "RE-НКА" или "RE-NFA") — это совокупность

- входного алфавита Σ,
- конечного множества состояний Q,
- множества начальных состояния $S \subseteq Q$,
- множество принимающих состояний $F \subseteq Q$,
- функция перехода $\delta : Q \times \mathrm{RE}(\Sigma) \to 2^Q$, что для всякого $q \in Q$ и всех $r \in RE(\Sigma)$ кроме, быть может, конечного множества $\delta(q, r) = \varnothing$.

Входными данными является конечная строка w. Входная строка принимается автоматом, если есть её разбиение на подстроки $w = u_1 \dots u_n$, последовательность регулярных выражений r_1, …, r_n и последовательность состояний q_0, …, q_n, что

- $u_k \in L(r_k)$ для всех k,
- $q_0 \in S$,
- $q_{k+1} \in \delta(q_k, r_{k+1})$,
- $q_n \in F$.

Язык автомата A или *язык, распознаваемый автоматом* A, — это множество $L(A)$ всех строк, распознаваемых (принимаемых) автоматом A. Язык, распознаваемый хоть каким-нибудь автоматом называется *регулярным*.

Замечание. Недетерминированные конечные автоматы с ε-переходами удобно изображать точно также как обычные НКА, но на стрелках (ориентированных рёбрах) переходов теперь можно писать не символы из Σ, а регулярные выражения из $\mathrm{RE}(\Sigma)$. Также по аналогии можно описать про существование пути в графе, но проще будет это вывести напрямик из уже данного определения.

Замечание. В этом случае также можно написать рекурсивную формулу δ^* (и через неё определить принимаемые слова). Но в этот раз она будет совсем муторной и бесполезной.

Лемма 11. *Всякий RE-НКА можно заменить на регулярное выражение, порождающее тот же язык.*

Доказательство. Заметим, что регулярное выражение для пустой строки есть $\varnothing^*$. Таким образом WLOG рассматриваемый RE-НКА автомат содержит одно начальное и одно принимающее состояние; так как иначе можно создать новые начальное и принимающее состояния, провести ε-переходы из нового начального в старые начальные состояния и из старых принимающих в новое принимающее и забыть роли старых начальных и принимающих состояний.

Будем уменьшать когда можно количество рёбер, а затем когда можно количество вершин. Рёбра будем склеивать кратные, т.е. если есть две вершины p и q два перехода из p в q по регулярным выражениям r_1 и r_2, то заменим на один переход из p в q по регулярному выражению $r_1|r_2$. Очевидно, что язык автомата не меняется. Таким образом кратных рёбер у нас исчезают.

Теперь пусть имеется состояние q отличное от начального и принимающего. Попытаемся удалить q. Для всяких вершин p_1 и p_2 рассмотрим регулярные выражения r_1, r_2 и r, соответствующие переходам из p_1 в q, из q в r_2 и из q в q соответственно, и заменим эти переходы (они будут удалены с удалением q) на перход из p_1 в p_2 по регулярному выражению $r_1 r * r_2$. Несложно проверить, что это тоже не изменит язык автомата.

Таким образом у нас останутся только два состояния и не более двух переходов между ними. Пусть регулярное выражение перехода от начального состояния к принимающему — r, а от принимающего к начальному — s. Следовательно, язык автомата порождается регулярным выражением $r(sr)^*$. □

2.2 Разные действия над автоматами

Определение 12. Пусть даны ДКА $A = (\Sigma, P, p_0, \eta, E)$ и $B = (\Sigma, Q, q_0, \delta, F)$. *Прямым произведением автоматов* A и B есть автомат $A \times B := C = (\Sigma, P \times Q, (p_0, q_0), \theta, E \times F)$, где

$$\theta((p, q), a) := (\eta(p, a), \delta(q, a)).$$

В таком случае $L(A \times B) = L(A) \cap L(B)$.

Также можно построить автомат $C = (\Sigma, P \times Q, (p_0, q_0), \theta, E \times Q \cup P \times F)$, где

$$\theta((p, q), a) := (\eta(p, a), \delta(q, a)).$$

В таком случае $L(C) = L(A) \cup L(B)$.

Также можно построить автомат $C = (\Sigma, P, p_0, \eta, P \setminus E)$. В таком случае $L(C) = \Sigma^* \setminus L(A)$.

Также можно построить автомат $C = (\Sigma, P^P, f_\varepsilon = \mathrm{Id}, \widehat{\varepsilon}, \{f \in P^P \mid f^2(q_0) \in F\})$, где P^P — множество функций $P \to P$, $f_w : P \to P$ — функция, переводящее всякое состояние p в состояние, которое получается из p прохождением по слову w, а

$$\widehat{\delta}(f, a) := f_a \circ f.$$

В таком случае $L(C) = \sqrt{L(A)} := \{w \in \Sigma^* \mid ww \in L(A)\}$.

Лемма 12 (о накачке, Рабин, Скотт)**.** *Пусть $L \subseteq \Sigma^*$ регулярен. Тогда есть константа $p \geqslant 1$, что для всякого слова w длины хотя бы p существует разбиение $w = xyz$, где $y \neq \varepsilon$ и $|xy| \leqslant p$, что для всех $l \geqslant 0$*

$$xy^l z \in L.$$

Пример 3. Язык $L = \{(ab)^n a^n\}_{n \in \mathbb{N}}$ не регулярен, но удовлетворяет лемме о накачке.