2.6. Конечные автоматы и регулярные выражения, их эквивалентность. Детерминизация и минимизация автоматов.

1. Краткая предыстория

Σ — множество символов (алфавит), Σ^* — множество слов, $L \subset \Sigma^*$ — язык.

Конечные автоматы и регулярные выражения — это такие штуки, которые распознают языки. Языки, для которых есть автомат, который их распознает называются регулярными.

2. Детерминированные конечные автоматы

Конечный автомат: (X, s, T, f). X — множество состояний, s — начальное состояние, $T \subset X$ — множество допускающих состояний, $f : X \times \Sigma \to X$ — функция переходов.

У конечного автомата есть текущее состояние q. В начале $q = s$. Он читает по одному символы слова и делает переходы $q \leftarrow f(q, c)$. Если в конце оказался в допускающем состоянии — значит слово принялось, если нет — значит не принялось...

3. Недетерминированные конечные автоматы

У этих может быть несколько переходов по одному и тому же символу. Слово допускается, если автомат **может** дойти до допускающего состояния. Еще там бывают ϵ-переходы, по ним можно переходить, не пропуская букву.

4. Регулярные выражения

Регулярные выражения задаются следующим образом. $x \in \Sigma$ — регулярное выражение, принимающее букву x. ϵ — регулярное выражение, принимающее пустую строку. AB — регулярное выражение, принимающее строки вида ab, где $a \in A$, $b \in B$ (конкатенация). $A|B$ — регулярное выражение, принимающее строки , которые принимает A или B (объединение). A^* — регулярное выражение, принимающее строки вида $a_1 a_2 ... a_k$, где все $a_i \in A$. Например, выражение $(0|1)^*$ принимает все строки из 0 и 1.

5. Эквивалентность

1. Построим по регулярному выражению недетерминированный автомат. Это просто.
2. Построим по недетерминированному автомату детерминированный. Возьмем за состояния нового автомата все подмножества состояний исходного автомата.

   Как строить переходы: возьмем текущее подмножество состояний. Посмотрим, куда из них можно попасть после получения буквы. В это подмножество и сделаем переход по этой букве.
3. Построим по детерминированному автомату регулярное выражение. Пусть $e(i, j)$ — регулярное выражение, описывающее все пути из состояния i в состояние j.

   Будем строить его как в алгоритме Флойда. На k итерации $e(i, j)$ учитывает все пути от i до j, у которых промежуточные состояния не больше k. На k итерации делаем $e(i, j) = e(i, j)|e(i, k)e(k, k)^*e(k, j)$. В конце получается то, что надо.

   Результирующее выражение получается объединением $e(s, t)$ для всех $t \in T$.

6. Минимизация автомата

Задача: найти автомат, принимающий тот же язык с минимальным числом состояний.

Чтобы минимизировать автомат, нужно слить эквивалентные состояния, то есть такие состояния, начав которых автомат принимает один и тот же язык.

Для этого будем последовательно генерировать пары не эквивалентных состояний по следующим принципам:

1. Если $a \in T$, а $b \notin T$, то a и b не эквивалентны.
2. Если $f(a, c) = a'$ и $f(b, c) = b'$ и a' и b' не эквивалентны, то a и b не эквивалентны.

Когда процесс успокоиться, все что не нашлись — эквивалентны. Сливаем их вместе — получаем минимизированный автомат.