1 ОТОБРАЖЕНИЯ БЕЗ ПРЕДВОСХИЩЕНИЯ

На вход вычислительного устройства обычно подаются тексты, представляющие собой последовательности символов. Набор используемых символов обычно называют *алфавитом.* Мы будем рассматривать только конечные алфавиты. Сами символы, из которых составлен алфавит, называют *буквами.*

Словом в рассматриваемом алфавите называют конечную последовательность букв этого алфавита, другими словами, отображение начального отрезка натуральных чисел в алфавит.

Если этот отрезок пуст, слово называется *пустым. Длина* пустого слова равна нулю. Мы будем пустое слово обозначать через λ. В противном случае, отрезок состоит из чисел 0,1, $\dots$, $n-1$ и *длина* слова равна n. Само это слово τ есть последовательность $\tau(0)\tau(1)\dots\tau(n-1)$. Через $|\tau|$ мы обозначаем длину слова τ. Через Σ^* обычно обозначают множество всех слов в алфавите Σ.

Вычислительное устройство перерабатывает информацию и, получив на вход некоторое слово, выдаёт на выходе некоторое другое слово. Это выходное слово зависит от того, что было подано на вход. Таким образом, вычислительное устройство задаёт *оператор*, перерабатывающий слова входного алфавита в слова выходного алфавита.

Этот оператор F каждому слову алфавита Σ ставит в соответствие некоторое однозначно определённое слово алфавита Δ, что обычно обозначают через $F:\Sigma^*\to\Delta^*$.

Оператор называют *синхронным*, если длина выходного слова равна длине входного слова. Мы будем рассматривать только синхронные операторы. Синхронный оператор, можно считать, работает дискретно и на каждом такте работы, считывая одну букву входа, выдаёт одну букву выхода.

Этот оператор F называют *детерминированным* или оператором без предвосхищения, если очередная буква выхода не зависит от букв входа, которые будут поданы позже этого такта работы. Точнее, i-ая буква $F(\tau)(i)$ слова $F(\tau)$ полностью определяется словом $\tau(0)\tau(1)\dots\tau(i)$. Мы будем рассматривать только детерминированные операторы.

Конкатенацией двух слов α и β называют такое слово γ, длина которого равна сумме длин слов α и β, что для $i = 0, 1 \dots, |\gamma| - 1$

$$\gamma(i) = \begin{cases} \alpha(i), & \text{если } i < |\alpha|, \\ \beta(i - |\alpha|), & \text{в остальных случаях.} \end{cases}$$

Другими словами, конкатенация двух слов — это слово, которое получится, если после первого слова сразу написать второе.

Например, конкатенацией слов "01110" и "110010" является "01110110010", а конкатенацией слов "пыле" и "сос" является "пылесос".

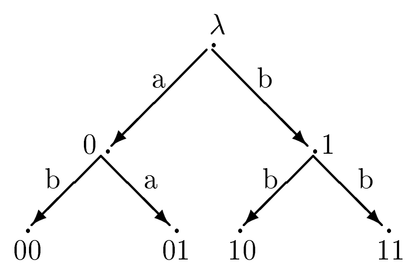
Конкатенацию слов ρ и τ обозначают через $\rho\tau$.

В этих терминах, для слова ρ *остаточным* оператором F_ρ оператора F называется такой оператор G, что для $i = 0, 1, \dots, |\tau|$

$$G(\tau)(i) = F(\rho\tau)(i - |\rho|).$$

Каждый детерминированный оператор может быть задан бесконечным нагруженным деревом следующим образом. Каждое ребро дерева помечено буквой, которая выдаётся, если идём по этому ребру. Из каждой вершины выходит столько рёбер, сколько букв имеется во входном алфавите. Эти рёбра упорядочены, соответственно упорядочению букв алфавита. Например, если алфавит состоит из двух букв, то левое ребро соответствует первой букве, а правое — второй. Каждая вершина соответствует остаточному оператору. Корень соответствует остаточному оператору для пустого слова. Если находимся в вершине, соответствующей остаточному оператору для слова ρ, и подаётся следующая буква a, то идём по ребру, соответствующему этой букве, и попадаем в вершину, соответствующую остаточному оператору для слова ρa.

Пример 1.0.1. *Начало нагруженного дерева, задающего детерминированный оператор.* Входные буквы — это 0 и 1. Выходные буквы — это a и b.



Видно, что остаточные операторы, соответствующие поданным на вход словам λ, "0" и "1", попарно различны.

Оператор F называется оператором *с конечной памятью*, если имеется такое конечное множество слов ρ_1, ρ_2, ... , ρ_m, что каждый остаточный оператор F_ρ оператора F совпадает с одним из операторов F_{ρ_1}, F_{ρ_2}, ... , F_{ρ_m}.

Пример 1.0.2.

Рассмотрим оператор, который в качестве входного алфавита использует буквы 0 и 1 и выдаёт 0, если в прочитанном слове чётное число нулей, и выдаёт 1, если в прочитанном слове число нулей нечётно. Ясно, что этот оператор имеет только два остаточных оператора, которые задаются словами с чётным и нечётным числами нулей.

Пример 1.0.3.

Рассмотрим последовательность
101001000100001 ... .

Каждый следующий блок нулей содержит на один нуль больше, чем предыдущий. Ясно, что оператор, выдающий, независимо от входа, эту последовательность, не имеет конечную память, так как все его остаточные операторы для входов разной длины различны.

Рассмотрим теперь оператор $F : \Sigma^* \to \Delta^*$ с конечной памятью. Рассмотрим такое такое конечное множество слов $\rho_1,\ \rho_2,\ \ldots\ ,\ \rho_m$, что каждый остаточный оператор F_ρ оператора F совпадает с одним из операторов $F_{\rho_1},\ F_{\rho_2},\ \ldots\ ,\ F_{\rho_m}$. Можно при этом предполагать, что остаточные операторы $F_{\rho_1},\ F_{\rho_2},\ \ldots\ ,\ F_{\rho_m}$ попарно различны.

Если для слова $\rho \in \Sigma^*$ и для некоторого $i \in \{1, 2, \ldots, m\}$ остаточный оператор F_ρ совпадает с остаточным оператором F_{ρ_i}, то в нагруженном дереве, задающем оператор F, заменим метку вершины, помеченной словом ρ, на ρ_i. Теперь многие разные вершины станут помечены одинаковыми метками. Назовём слова ρ_1, $\rho_2,\ \ldots\ ,\ \rho_m$ состояниями оператора F. Если при этом остаточный оператор F_λ совпадает с F_{ρ_1}, то состояние ρ_1 назовём начальным. Обозначим через Q множество $\{\rho_1,\ \rho_2,\ \ldots\ ,\ \rho_m\}$ всех состояний оператора F.

Пусть $(Q \times \Sigma) = \{(q, a) \mid q \in Q,\ a \in \Sigma\}$. Рассмотрим какую-то букву a из Σ и $i \in \{1, 2, \ldots, m\}$. Так как $F_{\rho_i a}$ совпадает с F_{ρ_j} для некоторого $j \in \{1, 2, \ldots, m\}$, то оператор F задаёт пару отображений $(Q \times \Sigma)$ в Q и в Δ. Первое отображение задаёт следующее состояние, другими словами, метку вершины помеченного дерева, задающего рассматриваемый оператор с конечной памятью, если из вершины с заданной меткой идти по ребру, соответствующему заданной букве входного алфавита. Второе отображение задаёт букву выходного алфавита, которая будет выдана на этом такте работы оператора.

Совокупность трёх конечных множеств Σ, Q и Δ, называемых *входным алфавитом*, *алфавитом состояний* и *выходным алфавитом*, вместе с выделенным элементом из алфавита состояний, который называется *начальным состоянием*, и отображением $(Q \times \Sigma)$ в $(Q \times \Delta)$ называется *конечным автоматом с выходом.*

Ясно, что зафиксировав в отображении состояние, мы получаем отображение из входного алфавита в $(Q \times \Delta)$. Если два разных состояния задают разные такие отображения, то такой конечный автомат с выходом мы будем называть *приведённым.*

Итак, каждый оператор с конечной памятью задаёт некоторый приведённый конечный автомат с выходом.

Верно и обратное. Имея приведённый конечный автомат с выходом, можно задать оператор с конечной памятью следующим образом.

Построим помеченное дерево, рёбра которого соответствуют буквам входного алфавита. Корень дерева метим начальным состоянием. Если начало ребра уже помечено состоянием q, а само ребро соответствует входной букве a, то конец ребра метим состоянием, которое отображение рассматриваемого конечного автомата с выходом сопоставляет паре (q, a), а само ребро метим выходной буквой, которая этим отображением сопоставляется паре (q, a).

Если этот приведённый конечный автомат с выходом был построен по оператору с конечной памятью, то в результате описанного построения мы снова получим первоначальный оператор с конечной памятью.

Ясно и то, что построив таким образом по приведённому конечному автомату с выходом оператор с конечной памятью, а потом по этому оператору снова построив конечный автомат, мы получим первоначально заданный конечный автомат с выходом.

Существенной особенностью работы конечного автомата является возможность регулярной (другими словами, правильной остановки). Дело в том, что некоторые входные слова обладают свойством завершённости. Например, текст программы может заканчиваться специальным символом, после чего можно приступать к выполнению этой программы. В конечном автомате это моделируется переходом в заключительное состояние. Итак, выделяется некоторое подмножество $Q_{\text{закл}}$ множества Q состояний рассматриваемого конечного автомата, а состояния из этого подмножества называются *заключительными*.

Как уже отмечалось, по состоянию и входной букве конечный автомат с выходом выдаёт новое состояние и выходную букву. На самом деле, обобщённым состоянием можно называть эту пару, состоящую из состояния и выходной буквы. В этом случае следующее обобщённое состояние зависит только от первого элемента пары и от входной буквы. Но это обстоятельство можно не принимать во внимание. Таким образом возникает понятие *конечного автомата*, которое и будет предметом нашего рассмотрения.