Формальные языки

Андрей Калмыков

Содержание

1 НКА и ДКА (и регулярки)

1.1 Регулярные выражения

Определение Регулярные выражения (Regular Expressions или RegEx) — формальный язык, используемый в компьютерных программах, работающих с текстом, для поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов (символов-джокеров, англ. wildcard characters).

Регулярки поддерживаются во многих языках программирования, а также во многих программах да и работают они довольно быстро. Проблема лишь в том, что регулярные выражения крайне нечитабельны, что породило массу шуток аля «Plural for regex is regrets». Поэтому, если вы будете использовать регулярные выражения вне формальных языков: например, в коде, то старайтесь либо объяснить назначение регулярки, чтобы человек, просматривающий ваш код не страшился написанных заклинаний.

```
preg_match('/^((([0-9A-Za-z]{1}[-0-9A-z\.]{1,}[0-9A-Za-z]{1})|([0-9А-Яа-я]{1}[-0-9А-я\.]{1,}[0-9А-Яа-я]{1}))@([-A-Za-z]{1,}\.){1,2}[-A-Za-z]{2,})$/u', $item)
```

Это, например, регулярное выражение для определения корректности e-mail'a, что выглядит довольно круто, но разбирать что-то подобное, написанное кем-то другим, — сущая пытка.

Теперь перейдём к регулярным выражениям с точки зрения формальных языков

Определение Рекурсивное определение регулярное выражения:

Reger R	**L(R)**
0	$\emptyset$
1	$\{\varepsilon\}$
$a : a \in \Sigma$	$\{a\}$
$R_1 + R_2$	$L_1 \cup L_2$
$R_1 \cdot R_2$	$L_1 \cdot L_2$
$R^x : x \in \mathbb{N}$	$L(R)^x$
R^+	$\bigcup_{i=1}^{\infty} L(R)^i$
R^*	$L(R^+) \cup \{\varepsilon\}$

Последние две операции называются соответственно **плюсом Клини** и **звездой Клини**
Первый листочек

1.2 НКА

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

Определение Формальный язык — некоторое подмножество Σ^*.

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

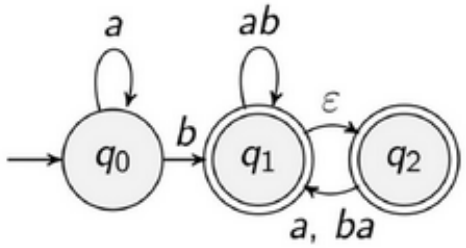
1. Q — множество состояний, Q — конечное множество, то есть $|Q| < \infty$;
2. Σ — алфавит;
3. $\Delta \subset Q \times \Sigma^* \times Q$ — множество переходов (т.е. состояние$_1$ $\overset{\text{слово}}{\longrightarrow}$ состояние$_2$);
4. $q_0 \in Q$ — стартовое состояние;

5. $F \subset Q$ — множество завершающих состояний.

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

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

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



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

Пример Рассмотрим, как автомат с картинки распознает слово $abab$:

$\langle q_0, abab\rangle \vdash \langle q_0, bab\rangle \vdash \langle q_1, ab\rangle \vdash \langle q_1, \varepsilon\rangle$

По транзитивности получаем: $\langle q_0, abab\rangle \vdash \langle q_1, \varepsilon\rangle$

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

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

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

Утверждение Для любого автоматного языка существует НКА, имеющий ровно 1 завершающее состояние

Доказательство Не будем приводить «формальное» доказательство, отметим лишь, что нужно добавить новое завершающее состояние, а потом ε-переходы в него. Доказательство эквивалентности двух автоматов тривиально.

Утверждение Для любого автоматного языка L существует НКА $M = \langle Q, \Sigma, \Delta, q_0, F\rangle$, такой что $L = L(M)$ и $\forall(\langle q_1, w\rangle \to q_2) \in \Delta \hookrightarrow |w| \leq 1$

Доказывается разбиением каждого ребра с большей длиной на рёбра длины 1.

Теперь рассмотрим усиленную версию этого утверждения:

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

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

Доказательство Достаточно доказать всего лишь корректность следующего алгоритма (так как курс совмещает в себе практическую и теоретическую часть, то заинтересованный ученик может попробовать доказать это своими силами ну или спросить у меня):

1. Убираем многократные ε-переходы

2. Переносим завершающие состояния
3. Добавляем транзитивные переходы
4. Убираем ε-переходы

Тут следует разбор примера: придумываем любой автомат с ε-переходами и избавляемся от них

1.3 ДКА

Определение НКА $M = \langle Q, \Sigma, \Delta, q_0, F \rangle$ называется *детерминированным (ДКА)*, если выполняются следующие условия:

1. $\forall (\langle q_1, w \rangle \to q_2) \in \Delta \hookrightarrow |w| = 1$ *(все переходы являются однобуквенными)*
2. $\forall a \in \Sigma, q \in Q \hookrightarrow |\Delta(q, a)| \leq 1$ *(из одного состояния по одному символу можно перейти не более, чем в одно состояние)*

Мотивация ДКА состоит в том, что мы хотим детерминированности в работе наших компиляторов. Мало кто хотел бы, чтобы его программа то компилировалась, то нет 😀

На самом же деле существует алгоритм перевода любого НКА В ДКА (и соответствующая теорема об эквивалентности множеств этих автоматов, которую мы опять опустим).

Алгоритм Томпсона:

1. Помещаем в очередь Q множество, состоящее только из стартовой вершины.
2. Затем, пока очередь не пуста выполняем следующие действия:
   - Достаем из очереди множество, назовем его q
   - Для всех $c \in \Sigma$ посмотрим в какое состояние ведет переход по символу c из каждого состояния в q. Полученное множество состояний положим в очередь Q только если оно не лежало там раньше. Каждое такое множество в итоговом ДКА будет отдельной вершиной, в которую будут вести переходы по соответствующим символам.
   - Если в множестве q хотя бы одна из вершин была терминальной в НКА, то соответствующая данному множеству вершина в ДКА также будет терминальной.

Пример использования алгоритма Томпсона

Переходим к решению второго листочка

Перед переходом к следующей теме, стоит вспомнить самое начало занятия, а точнее регулярные выражения. Существует теорема, которую мы фактически обойдём стороной в рамках данной пары, утверждающая, что класс автоматных языков совпадает с классом языком, задающихся регулярными выражениями. Эта теорема называется **Теоремой Клини** и доказывается в одну сторону путём построения регулярного выражения по автомату (в форме ДКА), а в другую сторону соответственно построением автомата по регулярному выражению. Доказательство довольно очевидно и каждый при желании может провести его сам.

2 ПДКА, МПДКА, лемма о разрастании

2.1 ПДКА, МПДКА и его построение

Мотивация Казалось бы, у нас уже достаточно видов конечных автоматов, но как проверить, что автоматы задают один язык? Понятно, что контрпример решает половину проблемы, но как быть с эквивалентными автоматами. Для решения данной проблемы введём ещё 2 определения.

Определение *Полным детерминированным конечным автоматом (ПДКА)* называется такой ДКА, что

$$\forall q \in Q, w \in \Sigma \exists q' \in \Sigma, \delta \in \Delta : \langle q, w \rangle \to q'$$

Для построения ПДКА из ДКА нужно всего лишь добавить новую вершину, которую будем называть «стоком», куда будут вести отсутствующие переходы из других состояний.

Заметим, что если мы хотим распознавать все слова, не принадлежащие языку, то нам нужно всего лишь инвертировать наш автомат — поменять терминальность/нетерминальность всех состояний.

Определение *Минимальным полным детерминированным конечным автоматом (МПДКА)* называется ПДКА минимальный по числу состояний.

Определение $q_1 \underset{n}{\sim} q_2$, если для любого слова $w : |w| \leq n \hookrightarrow$:

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

Кроме того введём соответствующее классы эквивалентности $Q \,/\, \underset{n}{\sim}$

Рассмотрим алгоритм построения МПДКА: Множество $Q \,/\, \underset{0}{\sim}$ представляет собой $\{F, Q \setminus F\}$, то есть это множество из множества завершающих состояний и множества состояний, не являющихся завершающими.

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

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

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

В соответствии с этим получим новое распределение состояний по классам

2.2 Лемма о разрастании (накачке)

Лемма Пусть L — автоматный язык, $|L| = \infty$. Тогда $\exists P \forall w \in L : |w| > P \hookrightarrow \exists x, y, z : w = xyz, |xy| \leq P, |y| \neq 0 : \forall k \in \mathbb{N} \hookrightarrow xy^k z \notin L$

Идея доказательства

1. Рассмотрим НКА с однобуквенными переходами M
2. Возьмём $P = |Q|$ и найдём первый цикл в M (он найдётся, т.к. $|w| > P$ по условию)

Теперь пусть часть слова до входа в цикл — x, сам цикл — y, а часть после — z. Докажем утверждения леммы:

1. $|xy| \leq P$ иначе мы должны были найти цикл раньше
2. $|y| \neq 0$ так как все переходы однобуквенные, а значит какое-то состояние не будет посещено из-за длины слова
3. $xy^k z \in L$ так как для любой степени k мы просто проходимся по циклу k раз $\square$

Зачастую мы будем пользоваться отрицанием лемму о разрастании, например, можем доказать, что язык $\{a^n b^n \mid n \in \mathbb{N}\}$ автоматным являться не будет (надо доказать, что если будем брать y в какой-то степени, то либо потеряется порядок, либо соотношение букв)

Далее решаем листочек на минимизацию автоматов и лемму о разрастании

3 Грамматики, КС грамматики, лемма о разрастании для КС грамматик

Мотивация Нам не хватает регулярок и автоматов, мы хотим распознавать более сложные конструкции и языки, придумаем что-нибудь новое

3.1 Грамматики

Так как мы фактически возводим новую теорию над старой, то наша новая теория в пределе будет включать нашу старую (как релятивистская механика включает в себя классическую)

Определение Слово w распознаётся грамматикой, если оно выводимо по правилам этой грамматики из S

Иерархия Хомского Пусть $A, B \in \mathbb{N}, \alpha, \varphi, \psi \in (N \cup \Sigma)^*, w \in \Sigma^*$. Тогда существуют следующие виды грамматик:

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

Как вы можете заметить каждая следующая грамматика расширяет предыдущую, уменьшая ограничения на правила и соответственно увеличивая количество распознаваемых языков. Определим основные грамматики

3.2 Порождающие грамматики

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

1. N — множество вспомогательных (нетерминальных) символов, $|N| < \infty$.
2. Σ — алфавит — множество (терминальных) символов, $|\Sigma| < \infty, \Sigma \cap N = \emptyset$
3. $S \in N$ — стартовый нетерминал
4. $P \in (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$ — множество правил, $|P| < \infty$

Определение Наименьшее рефлексивное транзитивное отношение $\vdash_G$ называется отношением выводимости в грамматике G, если

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

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

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

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

3.3 КС грамматики

Определение уже есть в табличке, так что приведём пример:

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

Примером не КС-языка будет $L = \{a^n b^n c^n\}$, но на данный момент нам не хватает аппарата для доказательства этого, спойлер: это просто будет лемма о разрастании для КС-грамматик, но к ней мы перейдём попозже

Заметим, что КС-языки замкнуты относительно объединения и конкатенации

До этого мы уже приводили автоматы к определённой форме, чтобы их можно было сравнивать, а также для удобства. Оказывается, что-то подобное можно провернуть и КС-грамматиками. Мы рассмотрим две нормальные формы КС-грамматик: Хомского и Грейбах, но для начала введём несколько определений.

Определение Символ $Y \in N$ называется *порождающим*, если $\exists w \in \Sigma^* : Y \vdash w$.

Определение Символ $D \in N$ называется *достижимым*, если $\exists \varphi, \psi \in (N \cup \Sigma)^* : S \vdash \varphi D \psi$.

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

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

Утверждение Для любой КС-грамматики существует эквивалентная ей (выводящая такой же язык), без бесполезных символов

Доказывается от противного

3.4 Нормальная форма Хомского

Определение КС-грамматика находится в нормальной форме Хомского, если все правила

имеют такой и только такой вид:

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

Чтобы привести грамматику к НФ Хомского нужно сделать следующие действия:

1. Удаление непорождающих символов
2. Удаление недостижимых символов
3. Удаление смешанных правил $D \to aBc$
   - Сделаем замену правила вида $A \to dBcEf$ на правила следующего вида: $A \to DBCEF$, $D \to d, C \to c, F \to f$.
4. Удаление длинных правил $A \to A_1 A_2 A_3 A_4$
   - Удалим длинные правила вида $B \to A_1 A_2 ... A_n$ с помощью замены на правила следующего вида:

$$B \to A_1 B_1, B_1 \to A_2 B_2 ... B_{n-1} \to A_{n-1} A_n$$

5. Удаление ε-порождающих символов
   - Теперь остались правила вида $A \to a, A \to BC, A \to B, A \to \varepsilon$. Если $A \to BC$ и $B \vdash \varepsilon$, то добавим $A \to C$. Затем удалим $A \to \varepsilon$
6. Обработка пустого слова
   - Может случиться, что $S \vdash \varepsilon$, тогда введём S' — новый стартовый нетерминал, $S' \to S$, если $S \vdash \varepsilon$, то $S' \to \varepsilon$ — новое правило
7. Удаление унарных (одиночных) правил $A \to B$
   - Здесь рассматриваем длинные унарные цепочки, если они приводят к двум нетерминалам или к терминалу, то просто удаляем промежуьточные звенья

3.5 Лемма о разрастании для КС грамматик

Возможно перейдёт на 4-й день

Лемма Пусть L — КС-язык. Тогда существует p, такое что для любого слова $w \in L$, длина

которого не меньше, чем p, существуют такие слова x, u, y, v, z, принадлежащие Σ^*, что $w = xuyvz, |uv| > 0, |uyv| \leq p$, что для любого $k \in \mathbb{N}$ выполняется, что $xu^k yv^k z \in L$. Доказывается здесь аналогично НКА, но мы пользуемся деревом разбора, а $p = 2^{|N|}$. *Разбор примера $L = \{a^n b^n c^n\}$ не КС*

4 Алгоритм CYK(КЯК), МП автоматы, образы языков

4.1 CYK

Определение Алгоритм Кока-Янгера-Касами позволяет по грамматике в нормальной форме (Хомского, но впоследствии мы поговорим и о форме Грейбах, а также о том, как это позволяет ускорить выполнение) вычислить, можно ли вывести данное слово в данной грамматике.

Если вдруг кто-либо из учеников знаком с алгоритмом Флойда-Уоршелла для поиска кратчайших путей, то вы можете заметить некую схожесть данных алгоритмов.

Идея алгоритма

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

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

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

1. $i = j$. Инициализируем массив для всех нетерминалов, из которых выводится какой-либо символ строки w. В таком случае $d[A][i][i] = \text{true}$, если в грамматике G присутствует правило $A \to w[i]$. Иначе $d[A][i][i] = \text{false}$.
2. $i \neq j$. Значения для всех нетерминалов и пар $\{\langle j', i'\rangle \mid j' - i' < m\}$ уже вычислены, поэтому

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

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

Если задуматься над теми действиями, что мы производим в алгоритме, то становится понятно назначение нормальной формы Хомского, потому что без подобных правил, алгоритм бы работал не совсем корректно.

Асимптотика Обработка правил вида $A \to w[i]$ выполняется за $O(n \cdot |P|)$. Проход по всем подстрокам выполняется за $O(n^2)$. В обработке одной подстроки присутствует цикл по всем правилам вывода и по всем разбиениям на две подстроки, следовательно, обработка работает за $O(n \cdot |P|)$. В итоге получаем конечную сложность $O(n^3 \cdot |P|)$.

Теперь поговорим о нормальной форме Грейбах. Она также определяется над Кс-грамматиками, имеет почти такие же требования к правилам, но вместо правил вида $A \to BC$ мы допускаем правила вида $A \to aB, A \to aBC$.

Данная нормальная форма позволяет, во-первых, доказать, что любой КС-язык можно прочитать с помощью МП-автомата, о которых мы поговорим чуть позже, а во-вторых, уменьшить константу в выше представленном алгоритме, так как каждый раз мы будем «откусывать» букву.

Раз данное ограничение называется нормальной формой Грейбах, то понятно, что и к нему можно привести любую КС-грамматику.

4.2 МП-автоматы

Идея До этого мы работали с автоматами, которые просто считывали слово и как-то работали по своей логике, но очеивдно, что ЭВМ имеют память, так почему же нашим автоматам её не иметь?

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

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

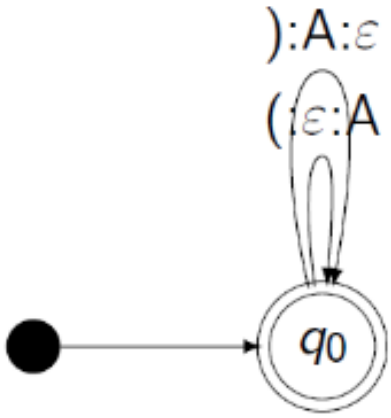
То есть мы снимаем что-то со стека, потом что-то читаем, а потом что-то кладём на стек

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

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

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

Пример на Правильные Скобочные Последовательности



Также очевидно, что любую операцию в МП автомате можно заменить на последовательность операций чтения без действий со стеком, взятия со стека и добавления на стек.
Классы КС-языков и МП-языков совпадают

Листочек по МП автоматам

5 Конечные преобразователи

Идея Все автоматы, с которыми мы работали до этого, позволяли ответить, принадлежит ли слово языку, но что делать, если мы хотим преобразовать слово по каким-то правилам?

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

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

Определение Конфигурация КПтеля T — это тройка $\langle q, u, v \rangle : q \in Q, u \in \Sigma^*, v \in \Gamma^*$

Неформально: находимся в состоянии q; осталось прочесть слово u; вывели слово v
Аналогично МП-автоматам определяется отношение выводимости
Определение Соответствие, задаваемое конечным преобразователем T:

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

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

Важно, чтобы ученики понимали, почему это не отображение
Примеры
1. $id(\Gamma = \Sigma)$, меняем на то же самое
2. Распознавание регулярных языков $\psi : \{(u, \varepsilon), u \in R\}$. Просто видоизменяем КА с помощью написания эпсилон на выход
3. Конкатенация также как с автоматами
4. Приписывание к слову чего-нибудь, что удовлетворяет регулярке. (Конкат из айди и регулярки)
5. Объединение как в автоматах
6. Гомоморфизм просто добавляем переход $x : \varphi(x)$

На КПтели есть очень классная теорема, которую вы возможно вспомните, когда будете проходить на тервере теорему Лебега(любую функцию распределения на $\mathbb{R}$ можно представить в виде суммы трёх функций распределения из конкретных классов)
Теорема Нива Любое конечное преобразование можно представить в виде композиции:
- обратного неудлиняющего гомоморфизма: $\psi^-1 : \Theta^* \to \Sigma$
- ограничения на регулярный язык: $id_R R \subset \Theta^*$
- неудлиняющего гомоморфизма: $\eta : \Theta^* \to \Gamma^*$

Листочек на КПтели
Листочки все есть, просто печатать их удобнее из отдельных файлов