Зимняя школа «Комбинаторика и алгоритмы - 2011»
Берендеевы поляны, 10-17 февраля

М.А.Ройтберг

«Конечные автоматы»

(12-14 февраля 2011)

А. СВОДКА ПОНЯТИЙ И РЕЗУЛЬТАТОВ.

Занятие 1

1.1. Алфавит. Слово. Пустое слово. Язык.

1.2. Конечный автомат. Конечный автомат допускает язык. Конечно-автоматный язык.

1.3. Эквивалентность состояний и автоматов. Отношение k-эквивалентности. Башня разбиений для отношение k-эквивалентности.

1.4. Утверждение 1. Пусть в автомате $T = <A, Q, F, q0, \varphi>$ n состояний; q', q'' – не эквивалентные состояния автомата T. Тогда состояния q' и q'' различаются некоторым словом длины не более $n-2$.
Пример, который показывает, что эту оценку нельзя улучшить.

Занятие 2

2.1. Любое конечное множество – конечно-автоматное (т.е. допускается конечным автоматом). Пустое множество - конечно-автоматное. Множество, состоящее из пустого слова – конечно-автоматное.

2.2. Если автомат с n состояниями допускает слово длины n, то он допускает слова как угодно большой длины.

2.3. Конкатенация (произведение) слов и языков. Языки частных. Вес языка.

2.4. Утверждение 2.

1. Пусть автомат T допускает язык L. Тогда вес языка L не превосходит количества состояний автомата T.

2. Пусть вес языка L равен n. Тогда существует автомат с n состояниями, допускающий язык L.

3. Такой автомат единственный.

Занятие 3

3.1. Замкнутость семейства конечно-автоматных языков относительно теоретико-множественных операций

3.2. Недетерминированные конечные автоматы. Языки, допускаемые недетерминированными конечными автоматами. Детерминизация. Конечно-автоматность языков, допускаемых недетерминированными конечными автоматами

3.3. Конкатенация и итерация. Замкнутость семейства конечно-автоматных языков относительно конкатенации и итерации.

3.4. Регулярные выражения. Регулярные языки. Конечно-автоматность регулярных языков.

3.5. Системы «линейных» уравнений над языками.

Утверждение 3.

1. Любой регулярный язык допускается некоторым конечным автоматом (см. п. 3.4).

2. Любой конечно-автоматный язык описывается некоторым регулярным выражением.

Б. РАСШИРЕННАЯ ПРОГРАММА

ЗАНЯТИЕ 1

1. Введение

1.0. О чем будет речь.

Вокруг нас много *текстов* – последовательностей символов . Наиболее знакомые нам тексты – в книгах. Они записаны буквами разных алфавитов (кириллица, латиница и много других). Но в текстах есть и другие знаки – цифры, знаки препинания, смайлики и др. Еще: пробелы, признаки конца строки, конца страницы и т.п. Т.е. на самом деле в текстах мы используем не 33 знака, а значительно больше [отступление в сторону: сколько?].

Формулы (не «двухэтажные»!) – тоже тексты. А «двухэтажные» формулы всегда можно записать, как тексты (например, используя знак '/' вместо дробной черты. Так делают в компьютерных программах.

[Текстами можно *кодировать* (не описывать!) изображения и звуки. Все, что хранится в компьютерах, - это тексты в двоичном алфавите. В сторону: их также можно рассматривать как тексты в алфавитах из 16 символов (16-чные цифры – группы по 4 двоичных символа), 256 символов (байты - группы по 8 двоичных символов). Иногда удобно в качестве символов группы из 16 и 32 двоичных символов].

Нас будут интересовать алгоритмы обработки текстов. Примеры: алгоритм решения квадратного уравнения; алгоритм, который записывает слово в обратном порядке; алгоритм, который в тексте всюду заменяет , на . ; алгоритм, который находит места, в которых отличаются два заданных текста; алгоритм вычисления длины текста. Все это алгоритмы обработки текстов.

Нас будут интересовать алгоритмы *распознавания (свойств)* текстов – алгоритмы, которые выдают (скажем) 1, если текст обладает некоторым свойством (=принадлежит некоторому множеству) и 0 в противном случае. Еще нас будут интересовать *конечные автоматы* – простейшие (модельные) устройства, реализующие алгоритмы распознавания текстов. Например – какие свойства текстов можно распознать с помощью конечных автоматов, а какие – нет.

1.1. Алфавит, слово, язык

Алфавит – (конечное!) множество *букв (=символов)*. Часто считается, что буквы в алфавите упорядочены («*алфавитный порядок*»).

Слово в алфавите *A* – конечная(!) последовательность символов в алфавите *A* .

Примеры (здесь и ниже слова, которые мы анализируем, выделены жирным; иногда их берут в кавычки):

1) **а, мама**

2) пустое слово (обозначается ε)

Множество всех слов в алфавите *A* (включая пустое слово) обозначается A^* (позже поговорим про это обозначение)

Длина слова – количество букв в нем. (длина пустого слова – 0).

Конкатенация – приписывание слов. Обозначается точкой (как умножение). Но это умножение не перестановочно ☺

Пример: **мама·ня** = **маманя**

Для любого слова w выполнено:

$$w \cdot \varepsilon = \varepsilon \cdot w = w$$

При конкатенации длины складываются.

Текст – это синоним терминов «слово», «символьная последовательность».
Неформально: текст – это длинное слово.

Язык – множество слов в алфавите A. Примеры:
1) пустой язык
2) язык $\{\varepsilon\}$, состоящий только из пустого слова;
3) язык, состоящий из заданного набора слов (конечный язык):
4) язык, состоящий из всех слов, который начинаются с данной буквы (бесконечный).

1.2. Задача распознавания языка.

Функция, которая распознает язык L в алфавите A, - это такая функция f, что

1) область определения f: A^*;
2) область значений f: $\{0, 1\}$;
3) $f(w)=1 \Leftrightarrow w \in L$

Будем рассматривать только функции, которые вычисляются с помощью некоторого устройства. Такие устройства бывают разных типов, эти типы называются *модели вычислений.*

В каждой модели вычислений есть:
1) входные данные,
2) промежуточные (рабочие) данные,
3) «процессор», который может находиться в одном из *конечного набора состояний.* В каждый момент времени процессор «видит» небольшую часть входных и промежуточных данных («одну ячейку»). Существует лишь конечное число вариантов видимой процессору ячейки данных.
4) *функция переходов,* которая определяет поведение процессора – в зависимости от
   - его состояния,
   - видимых процессору ячеек входных и рабочих данных.

Процессор работает по шагам. За один шаг можно
1) перейти в новое состояние,
2) записать что-то в текущую ячейку рабочих данных;
3) перейти к соседней ячейке рабочих данных;
4) перейти к соседней ячейке входных данных

Можно сделать действия всех этих типов!

Различные модели вычислений отличаются тем, как устроены входные и выходные данные. Примеры. Машина Тьюринга с 1 головкой и 1 лентой. Машина Тьюринга с плоской лентой. Машина Тьюринга с 2 головками и 1 лентой.
Задача распознавания симметрии слов.
Мащина Канеля с камущками (не разделены входные и рабочие данные).

2. **Конечные автоматы**

2.1. Определение

Конечный автомат – простейшая модель вычислений:
- рабочей памяти нет совсем;

- входные данные – слово;
- в начале смотрим на 1-ю (самую левую) на каждом шаге обязательно переходим к следующей букве;
- когда дошли до конца, работа заканчивается.

Результат (допускается ли входное слово) определяется по тому, в каком состоянии оказался автомат к концу работы.

Определение. Конечный автомат T – это пятерка $<A, Q, F, q0, \varphi>$, где

A – алфавит;

Q – множество состояний;

$F \subseteq Q$ - подмножество допускающих состояний;

$q0 \in Q$ – начальное состояние;

$\varphi: Q \times A \rightarrow Q$ – функция переходов

Автоматы удобно представлять в виде ориентированных графов (вершины – состояния, ребра изображают переходы, на каждом ребре изображена буква, соответствующая переходу). Из каждой вершины (состояния) для каждой буквы выходит ровно одно ребро, помеченное этой буквой. Над допускающими состояниями рисуем флажки.

Рисуем автоматы, спрашиваем - какие множества слов (языки) они распознают. По умолчанию, все автоматы в алфавите {0, 1}.

1) автомат с двумя состояниями, разные варианты
2) автомат с 3-мя состояниями, алфавит {0, 1, 2}

2.2. Построение автоматов. Технические приемы (тупики, дополнение)

1) автомат, который допускает множество слов, которые содержат:
   а) 1;
   б) 10;
   в) 101 (куда возвращаемся по 100?)

Задача 1 (самостоятельно). Построить автомат с алфавитом символов *{0, 1},* который допускает все слова, которые содержат 100101

2) автомат, который допускает множество слов, которые НЕ содержат 1 (тупиковое состояние);

3) автомат, который допускает множество слов, которые НЕ содержат 101 (альтернирование флажков = допускающих состояний). Автоматность дополнения.

4) Автомат, который распознает десятичные целые числа, которые делятся на 3.

Общая идея – есть некоторая характеристика начального отрезка слова, эта характеристика может принимать конечное количество значений. Допускается слово или нет, определяется значением этой характеристики. При чтении очередной буквы мы перевычисляем значение этой характеристики.

Еще примеры: автоматы, которые допускают:

1) пустое множество слов (т.е. автомат не допускает ничего);
2) множество, состоящее только из пустого слова;
3) отдельные буквы.

2.3. Вопросы на будущее:

1) какие множества нельзя распознать автоматом?
2) как минимизировать количество состояний?

3) единственность

3. Эквивалентность состояний и автоматов

3.1. Определение.

Состояния *q', q''* называются эквивалентными, если для любого слова *w* выполнено:

$$\varphi(q', w) \in F \Leftrightarrow \varphi(q'', w) \in F$$

Слово *w различает* состояния *q', q'',* если одно из состояний *φ(q', w), φ(q'', w)* – допускающее (т.е. лежит в *F),* а другое – нет.

3.2. Пример автомата с *n* состояниями и двумя состояниями, которые не различимы словом длины < *n-2*.

3.3. Доказательство верхней оценки длины различающего слова

Утверждение. Пусть в автомате $T = <A, Q, F, q0, \varphi>$ *n* состояний; *q', q''* – не эквивалентные состояния автомата *T*. Тогда состояния *q'* и *q''* различаются некоторым словом длины не более *n-2*.

Состояния *q', q''* называются *k*-эквивалентными, если для любого слова *w* длины не более *k* выполнено:

$$\varphi(q', w) \in F \Leftrightarrow \varphi(q'', w) \in F$$

Обозначение. *R(k)* - разбиение на множестве состояний, соответствующее отношению *k*-эквивалентности.

R(∞) - разбиение на множестве состояний, соответствующее отношению эквивалентности (без ограничения на длину различающего слова).

|*R*| - количество классов в разбиении *R*.

[Отступление. Понятие отношения эквивалентности (*a~a; a~b ⇔ b~a; a~b, b~c => a~c)*. Отношение эквивалентности, заданное на множестве *M,* разбивает это множество на классы эквивалентных элементов – *классы эквивалентности*]

Идея доказательства. Строим башню разбиений *R(0). R(1), …* .Слово "башня» означает, что каждое следующее разбиение – измельчение предыдущего. Или совпадает с ним. Если не измельчились – дальше измельчений не будет (это хорошо разобрать на примере).

Т.е., если *R(k) = R(k+1),* то

$$R(k) = R(k+1) = R(k+2) = R(k+3) = \ldots = R(\infty)$$

Пусть *k* – такое наименьшее число, что *R(k) = R(∞).* Нам нужно доказать, что *k≤n-2*.

Действительно, В *R(0)* – 2 класса (*F* и *Q-F*), т.е. |*R(0)*| =*2*. для всех *j=1, …, k* разбиения *R(j)* измельчаются, т.е.

$$|R(j)| \geq |R(j-1)| + 1 \ (j=1, \ldots, k)$$

. Поэтому

$$|R(k)| \geq |R(0)| + k = k+2.$$

Всего в разбиении не может быть больше классов, чем количество состояний *n*. Значит, *k+2 ≤n.* откуда *k ≤ n-2.* Ч.т.д.

ЗАНЯТИЕ 2

Вопросы с занятия 1:

какие множества нельзя распознать автоматом?
как минимизировать количество состояний?
единственность минимального автомата

1. Автоматы и конечные множества

1.1. Любое конечное множество – конечно-автоматное.

Пример 0. A = {а, м, н, о, я}

Язык: {**мама, маня, маманя, моня, няня, яма, яна}**

Состояния – начала слов (допускающие показаны жирным):

ε **(**пустое слово) – начальное состояние;

м,

ма,

мам, **мама**, маман, **маманя**

ман, **маня**,

мо, мон, **моня,**

н, ня, нян, **няня,**

я,

ям, **яма,**

ян, **яна**

1.2. Если автомат с *n* состояниями допускает слово длины *n,* то он допускает слова как угодно большой длины.

Доказательство. *Траектория* работы автомата (т.е. последовательность состояний, через которые он проходит во время работы над входным словом) на слове длины *n* имеет длину *n+1.* Поэтому одно из состояний в ней встречается дважды. И т.д.

2. Критерий конечно-автоматности языка. Минимальный автомат языка.

2.1. Конкатенация (произведение) слов и языков. Языки частных.

Пусть L, L1, L2 – языки; w – слово.

*L1*L2 = {xy | x ∈ L1 & y ∈ L2} - конкатенация*

*w*L = {wx | x ∈ L}*

L/w = {x| wx ∈ *L*} – *язык частных*

Примеры языков частных (всюду A = {0, 1})

1) L = 1*
   L/0 = ∅;.
   L/1 =L;
   Замечание: 1*L ≠ *L !!!!*.
2) Какие языки частных у языка L = {11111} ?
   Ответ:
   ∅; {1}, {11}, {111}, {1111}, {1111},

Утверждение 2.1. Слово *w*∈*L* ⇔ *ε* ∈ *L/w.*

2.2. Количество языков частных. Вес языка.

1) У каких языков есть ровно один язык частных?

Ответ: $\varnothing$; A*

2) Доказать, что во всех остальных случаях есть не менее 2-х языков частных.
3) Придумать бесконечный язык, у которого есть 3 различных языка частных.
   Ответ: *(01)**
   Тот же вопрос для 4-х языков, 5 языков и т.д.
4) Придумать язык, у которого бесконечное количество различных частных
   Решение: начальные фрагменты двоичной записи иррационального числа.

Определение. *Весом* языка *L* называется количество языков частных для языка *L*.

Определение. Пусть *L* – язык в алфавите *A*. Будем говорить, что слова *w1* и *w2* эквивалентны относительно языка *L* (синоним: *L*-эквивалентны), если:

$$L/w1 = L/w2$$

Пример. Пусть *L* – язык из примера 0. Тогда **маман, ман, нян –** *L*-эквивалентны *(L/x = {**я**})*, **ма, ня –** *L*-эквивалентны *(L/x = {**ня**})*, **мама** НЕ *L*-эквивалентно **ма** и **ня** *(L/**мама** = {ε, **ня**})*.

Замечание. 1. Классы эквивалентности для отношения *L*-эквивалентности находятся во взаимно-однозначном соответствии с языками частных языка *L*.

Утверждение 2.2. Пусть слова *w1, w2 L*-эквивалентны, $a \in A$. Тогда слова $w1 \cdot a$ и $w2 \cdot a$ тоже *L*-эквивалентны.

Доказательство. Пусть $x \in L/w1 \cdot a$. Тогда $w1 \cdot a \cdot x \in L \Rightarrow a \cdot x \in L/w1 = L/w2 \Rightarrow w2 \cdot a \cdot x \in L \Rightarrow x \in L/w2 \cdot a$, ч.т.д.

[полное понимание во время занятия всеми не требуется]

Утверждение 2.3. Пусть *L* – конечно-автоматный язык и *T* – распознающмй его автомат. Пусть после обработки слов *w1, w2* автомат *T* приходит в одно и то же состояние, т.е. $\varphi(q0, w1) = \varphi(q0, w2)$. Тогда слова *w1, w2* являются *L*-эквивалентными, т.е. $L/w1 = L/w2$.

Сколько частных может быть у конечно-автоматного языка, который допускается конечным автоматом с 3 состояниями?

ТЕОРЕМА. Пусть автомат *T* допускает язык *L*. Тогда вес языка *L* не превосходит количества состояний автомата *T*.

Доказательство. Пусть *n* – количество состояний автомата *T*. Допустим, что у языка *L* есть *n+1* различных языков частных: $L/w_1, \ldots, L/w_{n+1}$. Среди списка состояний $\varphi(q0, w_1), \ldots, \varphi(q0, w_{n+1})$ есть два одинаковых, пусть это $\varphi(q0, w_i)$ и $\varphi(q0, w_j)$. Тогда по утверждению 2.3, языки частных L/w_i и L/w_j совпадают. Противоречие.

Мы получили необходимое условие того, что язык является конечно-автоматным.

Задача. Привести примеры языков, которые ек являются конечно-автоматными.

Вопрос: является ли это условие достаточным?

3. Канонический автомат для конечно-автоматного языка.

3.1. Конструкция.

Пусть L – язык конечного веса, *вес(L) = n.* Канонический автомат для языка L – это автомат T, который строится следующим образом.

Множество состояний автомата T – это множество языков частных языка L. Таким образом, каждому состоянию соответствует класс L-эквивалентности.

Начальное состояние – язык $L = L/\varepsilon$. Функция переходов φ: пусть $M = L/w$ – язык частных языка L, порожденный некоторым словом w (и любым словом, которое L-эквивалентно слову w). Пусть a – буква. Тогда

$$\varphi(M, a) = M/a = L/wa$$

Заключительные состояния: языки частных, содержащие пустое слово.

3.2. Канонический автомат – минимальный автомат, допускающий L. Единственность.

Теорема.

1. Построенный автомат T допускает язык L.
2. Если автомат T' допускает язык L и содержит столько же состояний, сколько автомат T, то автомат T' эквивалентен автомату T (т.е. между состояниями T и T' есть взаимно-однозначное соответствие, согласованное с функцией переходов, причем начальные состояния соответствуют друг другу, а заключительное состояние одного автомата соответствует заключительному состоянию другого).

Доказательство (набросок).

П.1 (а)Индукцией по длине слова w доказывается, что

$$\varphi(L, w) = L/w$$

(b) Очевидное утверждение:

$$w \in L \Leftrightarrow \varepsilon \in L/w$$

Далее – ясно.

П.2. Каждому состоянию автомата T' соответствует язык частных языка L (см. утверждение 2.3). Т.к. количество состояний автомата T' равно количеству языков частных, то между состояниями автомата T (=языками частных) и состояниями автомата T' есть взаимно-однозначное соответствие. Нужно показать, что это соответствие согласовано с функциями переходов, что начальные состояния T и T'соответствуют друг другу, а заключительное состояние одного автомата соответствует заключительному состоянию другого. Это примерно понятно, технические детали не приводятся.

Пример 1.

{а, м, н, о, я}

мама, маня, моня, яма, яна, маманя

Пример 2. (012)* + 33

ЗАНЯТИЕ 3

1. Замкнутость семейства конечно-автоматных языков относительно теоретико-множественных операций, конкатенации и итерации.

1.1. Теоретико-множественные операции.

Напоминание: «Простые» множества (пустое множество; множество, состоящее только из пустого слова; любое конечное множество) – конечно-автоматные.

Дополнение конечно-автоматного множества – конечно-автоматное («альтернирование флажков», т.е. $F' = Q-F$).

Как обстоит дело с пересечением?

Пусть $L1$, $L2$ – конечно-автоматные языки, которые допускаются соответственно автоматами $T1 = <A, Q1, F1, q1, \varphi1>$ и
$T2 = <A, Q2, F2, q2, \varphi2>$.

Декартово произведение автоматов $T1$ и $T2$ – это автомат
$T = <A, Q1 \times Q2, F1 \times F2, (q1, q2), \varphi>$,
где
$\varphi((q', q''), a) = (\varphi1(q', a), \varphi2(q'', a))$

Утверждение. Автомат T допускает пересечение множеств $L1$ и $L2$.

Как с объединением?

Рассмотрим автомат T', который отличается от $T = T1 \times T2$ только множеством допускающих состояний:

$F' = F1 \times Q2 \cup Q1 \cup F2$

Утверждение. Автомат T' допускает объединение множеств $L1$ и $L2$.

Замечание. $F1 \times Q2 \cup Q1 \cup F2 = (Q1 \times Q2) - ((Q1-F1) \times (Q2-F2))$. Это соответствует известным теоретико-множественным соотношениям ($V1$, $V2$ – подмножества множества U):

$$V1 \cup V2 = U - ((U-V1) \cap (U-V2))$$

1.2. Конкатенация и итерация

Новые операции и обозначения.

Определение.

Степени языков:

$L^2 = L \cdot L$; $L^3 = L^2 \cdot L$ и т.д.

$L^0 = \{\varepsilon\}$

Итерация языка L – это объединение всех степеней языка L.

Обозначение: L^*.

Примеры.

Всегда ли можно ли построить автомат, допускающий конкатенацию конечно-автоматных языков?

Пример 1. $L1 =$ {**мама, папа**}; $L2 =$ { **сын, дед} .**

Решение – «конкатенация автоматов»: все стрелки, которые ведут в допускающее состояние $T1$, перенаправляем в стартовое состояние автомата $T2$.

Пример 2. $L1 =$ {**мама, маманя**}; $L1 =$ { **няня, дочь} .**

Найденная конструкция не работает, т.к. теряются слова, которые начинаются с **маманя.**

Неформально: для состояния **мам** автомата *T* нужны две выходящие стрелки, соответствующие букве **а**: одна будет вести в состояние **мама** автомата *T1,* а другая – в стартовое состояние автомата *T2.*

С итерацией ситуация аналогичная. Автомат, допускающий итерацию *{***мама, папа***}** строится легко, а для итерации *{***мама, маманя***}** потребуются две стрелки для одной буквы.

С этими кратными стрелками сейчас будем разбираться.

2. Недетерминированные автоматы. Детерминизация.

2.1. Определение.

Говоря неформально, *недетерминированный* конечный автомат – это ориентированный граф, «стрелки» которого помечены буквами выбранного алфавита *A,* а вершины соответствуют состояниям. В отличие от *детерминированных* конечных автоматов, которые мы рассматривали раньше, НЕ ТРЕБУЕТСЯ, чтобы из каждого состояния для каждой буквы была только одна выходящая стрелка. Таких стрелок может быть много, а может не быть вообще. Как обычно, выделено множество заключительных состояний. А начальных состояний может быть более одного.

Таким образом, каждому слову может соответствовать много путей в графе автомата (НАПОМИНАНИЕ: путь соответствует слову, если (1) он начинается в начальном состоянии автомата и (2) пометки на стрелках образуют данное слово). Будем говорить, что недетерминированный автомат *допускает* слово, если хоть один такой путь ведет в допускающее состояние.

Определение. Недетерминированный конечный автомат *T* – это пятерка $<A, Q, F, Q0, \Phi>$, где

A – алфавит;
Q – множество состояний;
$F \subseteq Q$ - подмножество допускающих состояний;
$Q0 \subseteq Q$ – непустое подмножество начальных состояний;
$\Phi \subseteq Q \times A \times Q$ – множество (разрешенных) переходов

Пример. Язык (01)*00 - недетерминированный допускающий автомат, построенный как конкатенация автоматов для (01)* и 00.

2.2. Детерминизация.

Утверждение. Пусть $T = <A, Q, F, Q0, \Phi>$ - недетерминированный конечный автомат и *L* – язык, который допускает автомат *T.* Тогда существует детерминированный автомат, который допускает тот же язык *L.*

Доказательство (набросок). Рассмотрим в качестве искомого детерминированного автомата, автомат, определенный таким образом:

1) множество состояний: 2^Q (т.е. множество всех подмножеств множества *Q*);

2) начальное состояние: множество $Q0 \in 2^Q$;

3) конечными состояниями являются все такие множества $Z \subseteq Q$, которые пересекаются с множеством *F;*

4) функция переходов φ: пусть $R \in 2^Q$ (т.е. *R* - подмножество *Q), a* $\in A$. Тогда $\varphi(R, a) = \{q \mid \exists\, q' \in R: <q', a, q> \in \Phi\}$

Говоря неформально, при работе над словом *w* построенный автомат “помнит» все состояния, в которых оканчиваются пути, соответствующие слову *w*.

Это можно доказать с помощью метода математической индукции по длине слова *w*. И т.д.

2.3. Пример.

Построение детерминированного автомата, распознающего язык (01)*00:

1) с помощью детерминизаци;
2) с помощью языков частных

$L/\varepsilon = L;$

$L/1 = \varnothing;$

$L/0 = 1 \cdot L + 0;$

$L/00 = (L/0)/0 = \{ \varepsilon \}$

$L/01 = L$

Состояния: $L;\ L/0 = 1 \cdot L + 0;\ L/00 = \{ \varepsilon \};\ \varnothing$ = тупик.

Переходы – очевидно.

3. **Регулярные выражения.**

3.1. Определения.

Регулярные операции над языками:

1) объединение;
2) конкатенация;
3) итерация.

Базовые регулярные языки:

1) пустой язык;
2) язык, состоящий только из пустого слова;
3) язык, состоящий из одного однобуквенного слова («однобуквенные») языки.

Регулярные языки – языки, которые можно получить из базовых с помощью регулярных операций.

Регулярные языки можно описывать с помощью *регулярных выражений.* А именно:

$\varnothing$ обозначает пустой язык;

ε обозначает язык, состоящий только из пустого слова;

a обозначает язык, состоящий из одного однобуквенного слова *'a' ($a \in A$);*

(L1+L2) обозначает объединение языков *L1* и *L2;*

(L1·L2) обозначает конкатенацию языков *L1* и *L2;*

*(L)** обозначает итерацию языка *L;*

Как и в арифметике, будем считать, что «умножение» связывает аргументы сильнее, чем «сложение». Поэтому *QR+S* будет означать *(Q·R)+S.*

Примеры регулярных выражений.

Задачи вида: лежит ли данное слово в данном регулярном выражении?

3.2. Любой регулярный язык – конечно-автоматный

Утверждение. Любой регулярный язык допускается некоторым конечным автоматом.

Доказательство. Индукцией по построению регулярного языка. Мы раньше доказали, что

а) базовые регулярные языки – конечно-автоматные;

б) если языки *L1, L2* – конечно-автоматные, то языки, которые можно получить из них с помощью регулярных операций – конечно-автоматные.

Верно ли обратное?
Придется рассмотреть системы линейных уравнений над языками с регулярными коэффициентами.

3.3. Уравнение с одним неизвестным.

Рассмотрим уравнение (X – неизвестный язык, α и β – известные языки, т.е. коэффициенты уравнения):

$$X = \alpha X + \beta \qquad (1)$$

Утверждение.

1. Язык

$$X0 = (\alpha^*)\beta \qquad (2)$$

является решением уравнения (1).

2. Если X' – решение (1), то $X0 \subseteq X'$.

3. Если α не содержит пустого слова, то $X0 = (\alpha^*)\beta$ – единственное решение (1).

Доказательство (набросок, не трогаем п.3). Имеем: $\beta \subseteq X \Rightarrow \alpha\beta \subseteq X \Rightarrow \alpha^2\beta \subseteq X$. И т.д.

Язык $(\alpha^*)\beta$ будем называть *минимальным* решением уравнения (1).

Вопрос. Почему мы записываем уравнение в виде

$$X = \alpha X + \beta$$

а не в виде

$$\alpha X + \beta = 0$$

как в обычной алгебре?

Ответ: потому, что у операции объединения (которая у нас обозначается знаком +) нет обратной. Это хороший вопрос, но, чтобы полностью ответить на него, нужно углубляться в алгебру.

3.4. Системы линейных уравнений над языками (на примере 2-х неизвестных).

Общий вид:

$$X = \alpha_{1,1}X + \alpha_{1,2}Y + \beta_1 \qquad (3)$$

$$Y = \alpha_{2,1}X + \alpha_{2,2}Y + \beta_2 \qquad (4)$$

Подход к решению: исключение неизвестных (вспомним, как решать обычные линейные системы). Из (4):

$$Y = \alpha_{2,2}Y + (\alpha_{2,1}X + \beta_2)$$

Из (2):

$$Y = (\alpha_{2,2})^*(\alpha_{2,1}X + \beta_2) = (((\alpha_{2,2})^*)\alpha_{2,1})X + ((\alpha_{2,2})^*)\beta_2 \qquad (5)$$

Подставляем в (3). И т.д.

!!! Для конкатенации и объединения действуют распределительный и сочетательный законы. Поэтому скобки можно раскрывать как обычно.

Утверждение.

1. При любых коэффициентах система (3)-(4) имеет решение.

2. Среди этих решений есть минимальное решение, т.е. такое, которое содержится в любом другом решении (напоминаем: решение системы (3)-(4) – пара множеств).

3. Минимальное решение можно найти с помощью метода исключения переменных (см. ниже).

4. Если множества $\alpha_{1,1}$ и $\alpha_{2,2}$ не содержат пустого слова, то минимальное решение – единственное.

Доказательство – примерно ясно. Не приводится.

Утверждение. Если коэффициенты системы – регулярные языки, то и минимальное решение – регулярный язык.

Доказательство – ясно.

3.5. Системы линейных уравнений над языками и автоматы

По конечному автомату T можно построить систему линейных уравнений. Неизвестные соответствуют состояниям. Коэффициенты пр неизвестных в правых частям – пометкам на стрелках в графе (переходам). Если в левой части состояние не допускающее, то свободного члена нет. В противном случае свободный член – это язык, состоящий из пустого слова.

Утверждение. Язык, допускаемый конечным автоматом T, совпадает с минимальным решением системы уравнений, соответствующей этому автомату.

Следствие. Любой конечно-автоматный язык – регулярный.

Доказательство – не приводится (примерно ясно).

Упражнение. Построить систему уравнений по данному недетерминированному автомату.