Unity. Precision. Perfection.

КОНСПЕКТ ЛЕКЦИЙ

по дисциплине «Теория автоматов и формальных языков»

Лектор: Вербицкая
Страниц: 17
Последнее обновление: 18 сентября 2019 г.
Автор: Корытов Павел, 6304

Санкт-Петербург
2019

Содержание

1 Введение **2**

1.1 Синтаксис и семантика 2

1.2 Что такое язык? Теория множеств 2

1.3 Формальный язык 4

1.3.1 БНФ — Бэкуса-Наура форма 5

1.3.2 Расширенная БНФ (EBNF) 5

1.3.3 Синтаксические диаграммы Вирта 6

1.4 Формальная грамматика 6

2 Конечные автоматы **11**

2.1 Теоретико-множественное доказательство невозможности описания языков 11

2.2 Конечные автоматы 11

2.2.1 Определение 12

2.2.2 Путь в конечном автомате 13

2.2.3 Распознавание слова конечным автоматом 14

2.3 Эквивалетность конечных автоматов 14

2.3.1 Классы эквивалетности 15

2.3.2 Эквивалетные состояния 15

2.3.3 Алгоритм минимизации 15

2.4 Недетерминированный КА 16

1. Введение

Виды языков:

- Естественные — Русский, английский
- Искуственные
  - Эсперанто, ложбан
  - Клингонский, эльфийский
  - C++, Python, Java, C#, Haskell, OCaml, Perl, Coq, Agda

1.1. Синтаксис и семантика

- *Синтаксис* — правила построения программ из символов
- *Семантика* — правила истолкования программ

Пример 1.1. Язык арифметических выражений

$$1 \bullet (2 + 3)/4 - 5$$

- Синтаксис
  - *Терм* — последовательность цифр или любое выражение в скобках
  - *Слагаемое* , последовательность *термов* , соединенных знаками умножения и деление
  - *Выражение* — последовательность *слагаемых* , соединенных знаками сложения и вычитания (перед первым слагаемым может стоят минус)
- Семантика
  - Значение арифметического выражения
    - -3.75
    - -4
    - $\frac{-15}{4}$

1.2. Что такое язык? Теория множеств

Язык — множество строк

Множества

Множество — набор уникальных элементов

- $x \in X$: x — элемент множества X (x принадлежит X)
- $x \notin X$ — x не являеся элементом X
- Уникальность, неупорядоченность: $\{13, 42\} = \{42, 13\} = \{13, 42, 13\}$
- Универсальное множество (универсум) — множество всех мыслимых объектов
  - $[N] = \{1, 2, 3, \ldots\}$
  - $[Z] = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
  - $[Q] = m/n | m, n \in [Z]; n \neq 0$

Подмножества

A является *подмножеством* B тогда и только тогда, когда все элементы A являеются элементами B.

$$A \subseteq B \iff \forall x : x \in A \Rightarrow x \in B$$

Пустое множество ($\emptyset$) — мноэество без элементов

- $\forall x : x \notin \emptyset$
- $\forall A : \emptyset \subseteq A$

Множества A и B *равны* тогда и только тогда, когда A является подмножеством B и B является подмножеством A

$$A = B \iff (A \subseteq B) \wedge (B \subseteq A)$$

A является *строгим подмножеством* B тогда и только тогда, когда A является подмножеством B, но они не равны друг другу

$$A \subset B \iff (A \subseteq B) \wedge A \neq B$$

- $\forall x : \emptyset \subset \{x\}$
- $\forall A : (A = A) \wedge A \not\subset B$

$2^A = \{B | B \subseteq A\}$ — *powerset*, множество всех подмножеств A

- $\forall A : \emptyset in 2^A$
- $\forall A : A \in 2^A$
- $A = \{0, 1\} \Rightarrow \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$

Операции над множествами

- *Объединение* : $A \cup B = \{x|(x \in A) \vee (x \in B)\}$
- *Пересечение* : $A \cap B = \{x|(x \in A) \wedge (x \in B)\}$
- *Разность* : A
  $B = x|(x \in A) \wedge (x \notin B)$
- *Дополнение* : $\bar{A} = x|(x \in u) \wedge (x \notin A) = u$
  A

Строки

Строка — последовательность символов

- *Алфавит* (Σ) — конечное множество (атомарных, неделимых символов)
- *Цепочка* — любая конечная последовательность символов алфавита — предложение, слово, строка, …
  ε — цепочка, не содержащая ни одного символа

Операции над строками

- Конкатенация строк α и β ($\alpha \bullet \beta = \alpha\beta$) — соединение строк
  - $\forall\alpha\beta\gamma : (\alpha \bullet \beta) \bullet \gamma = \alpha \bullet (\beta \bullet \gamma)$
  - $\forall\alpha : \alpha \bullet \varepsilon = \varepsilon \bullet \alpha = \alpha$
- *Обращение (реверс)* a^R — цепочка, символы которой записаны в обратном порядке
- n-я степень цепочки a^n — конкатенация n повторений цепочки
- $|a|$ — *длина строки* — количество составляющих её символов

Пример — арифметические выражения
Алфавит $\Sigma = \{0, 1, \ldots, 9, +, -, *, /, (,)\}$

1.3. Формальный язык

- Σ — алфавит
  - $\Sigma = \{0, 1\}$
- Σ^* — подмножетсов, содержаещее все цепочке в алфавите Σ, включая пустую цепочку
- $\Sigma^+ = \Sigma^*/\{\varepsilon\}$
- Формальный язык в алфавите Σ — подмножество множества всех цепочек в этом алфавите

- Для любого языка L (в алфавите Σ) справедливо $L \subseteq \Sigma$
- $L = \{0, 0, 000, \ldots\} \subset \{0, 1\}^*$
- $L = \{0, 0101, \ldots\} \subset \{0, 1\}^*$

Метаязык — язык, на котором дано описание языка

- Естественный язык
- Язык металингвистических формул Бэкуса (БНФ)
- Синтаксические диаграммы
- Грамматики
- …

1.3.1. БНФ — Бэкуса-Наура форма

- *Символ* — элементарное понятие языка
  - $+$ — означает сложение в языке арифметических выражений
- *Метапеременная* — сложное понятие языка
  - Переменной <выражение> можно обозначить выражение
- *Формула*
  - <определяемый символ> := <посл 1> | … | <посл n>
  - В правой части формулы — альтернатива конкатенаций строк, составленных из символов и метапеременных
- Пример: число
  - <цифра> := | 0 | 1 | … | 9
  - <число> := <цифра>|<цифра><число>

1.3.2. Расширенная БНФ (EBNF)

- *Итерация*
  - $< x >:= \{< y >\}$ эквивалентно $< x >:= \varepsilon | < y >< x >$
- *Условное вхождение*
  - $< x >:= [< y >]$ эквивалентно $< x >:= \varepsilon | < y >$
- *Скобки для группировки*
  - $(< x > | < y >) < z >$ эквивалентно $< x >< z > | < y >< z >$

Пример 1.2. Арифметические выражения

$$< expr >:= [-] < factor > \{(+|-) < factor >\}$$

$$< factor >:=< term > \{(*|/) < term >\}$$

$$< term >:=< number > /'('< expr >')'$$

1.3.3. Синтаксические диаграммы Вирта

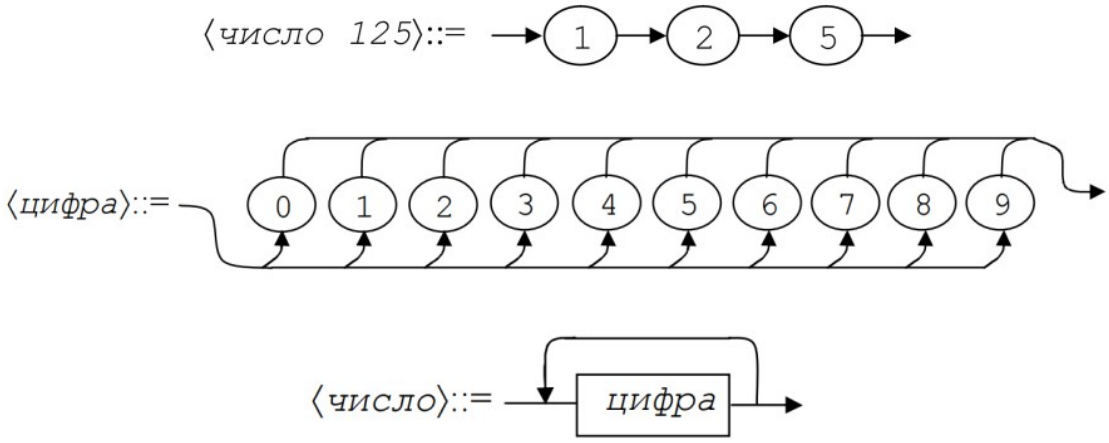


Рисунок 1. Синтаксические диаграммы Вирта

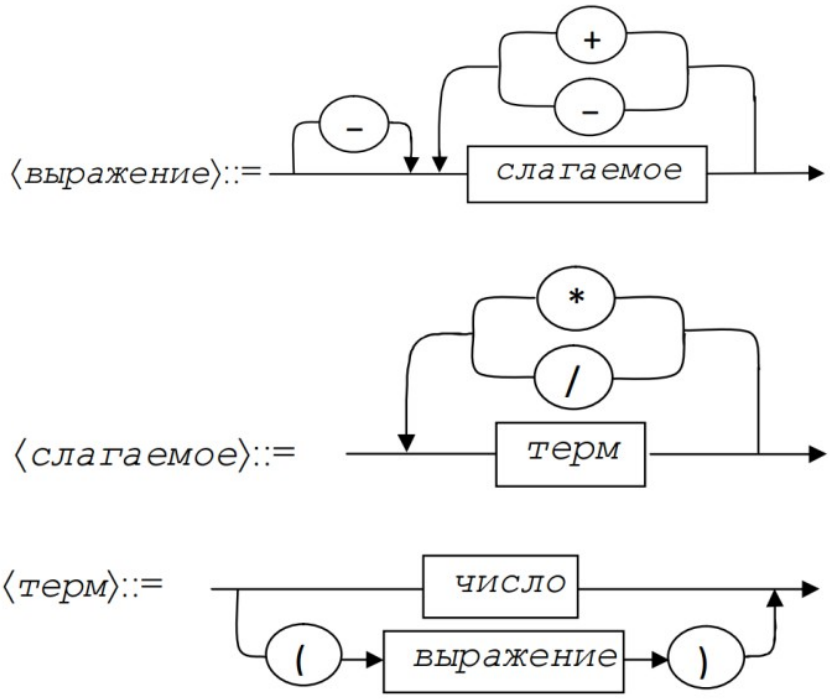


Рисунок 2. Синтаксические диаграммы Вирта

1.4. Формальная грамматика

- *Порождающая грамматика* G — это четверка $\{V_T, V_N, P, S\}$

- V_T — алфавит терминальных символов (*терминалов*)
- V_N — алфавит нетерминальных символов (*нетерминалов*)
  - $V_T \cap V_N = \emptyset$
  - $V : +V_T \cap V_N$
- P — конечное множество правил вида $\alpha \to \beta$
  - $\alpha \in V^* V_N V^*$
  - $\beta in V^*$
- S — начальный нетерминал грамматики
  - $S \in N$

Пример 1.3. Язык чисел в двоичной системе счисления

$$V_T = \{0, 1, -\}; V_n = \{S, N, A\}$$

- $S \to 0$
- $S \to N$
- $S \to -N$
- $N \to 1A$
- $A \to 0A$
- $A \to 1A$
- $A \to \varepsilon$

После преобразований:

- $S \to 0 \mid N \mid -N$
- $N \to 1A$
- $A \to 0A \mid 1A \mid \varepsilon$

Или так:

- $S \to 0 \mid [-]N$
- $N \to 1A$
- $A \to (0 \mid 1)A \mid \varepsilon$

Отношение непосредственной выводимости

- $\alpha \to \beta \in P$
- $\gamma, \delta \in V^*$
- $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$: $\gamma\beta\delta$ *непосредственно выводится* из $\gamma\alpha\beta$ при помощи правила $\alpha \to \beta$

Отношение выводимости — рефлексивно-транзитивное замыкание отношения непосредственной выводимости

- $\alpha_0, \alpha_1, \ldots, \alpha_n \in V^*$
- $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_n$
- $\alpha_0 \overset{*}{\Rightarrow} \alpha_n : \alpha_n$ *выводится из* α_0

Пример 1.4. [
Отношение выводимости]

$$S \to 0|N| - N$$

$$N \to 1A$$

$$A \to 0A|1A|\varepsilon$$

$$S \Rightarrow -N \Rightarrow -1A \Rightarrow -11A \Rightarrow^* -1101A \Rightarrow -1101$$

Свойства отношения выводимости

- Транзитивность
  $\forall \alpha, \beta, \gamma in V^* : \alpha \overset{*}{\Rightarrow} \beta, \beta \overset{*}{\Rightarrow} \gamma$ следовательно $\alpha \overset{*}{\Rightarrow} \gamma$
- Рефлексивность: $\forall \alpha \in V^* : \alpha \overset{*}{\Rightarrow} \alpha$
- $\alpha_0 \overset{+}{\Rightarrow} \alpha_n$: вывод использует хотя бы одно правило грамматики
- $\alpha_0 \overset{k}{\Rightarrow} \alpha_n$: вывод происходит за k шагов

Левосторонний вывод

На каждом шагу заменяем самый левый нетерминал

$$S \to AA \mid s$$

$$A \to a$$

$$A \to AA \mid Bb \mid a$$

$$B \to c \mid d$$

$$S \Rightarrow AA \Rightarrow BbA \Rightarrow cbA \Rightarrow cbAA \Rightarrow cbaA \Rightarrow cbaa$$

Правосторонний вывод определяется аналогично

Язык, порождаемый грамматикой

— всевозможные цепочки из тервиналов, которые выводятся из стартового терминала

$$L(G) = \{\omega \in V_T^* | S \overset{*}{\Rightarrow} \omega\}$$

Грамматики G_1 и G_2 эквивалентны, если $L(G_1) = L(G_2)$

Контекстно-свободная грамматика

— грамматика, все правила которой имеют вид $A \to \alpha, A \in V_n, \alpha \in V^*$. В левой части находится только один терминал

Проблема такой грамматики в том, что запись естественных языков (и некоторых языков программирования) в таком виде невозможожна. Например, значение оператора $*$ в C++ зависит от контекста. В Java же синтаксис контекстно-свободный.

Тем не менее, с помощью контестно-свободной грамматики можно записать отдельные элементы языков.

Дерево вывода

Дерево является *деревом вывода* для $G = \{V_n, V_T, P, S\}$, если

- Каждый узел помечен символов из алфавита V
- Метка корня — S
- Листья помечены терминалами, остальные узлы — нетерминалами
- Если узлы $n_0, \ldots, n_k$ — прямые потомки узла n, перечисленные слева направо, с метками

Пример 1.5. Дерево вывода
$G =< \{S, A\}, \{a, b\}, \{S \to aAS \mid a, A \to SbA \mid ba \mid sSS\}, S >$

$S \Rightarrow aAS \Rightarrow aSBaS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$

Теорема. Пусть $G = \{V_N, V_T, P, S\}$ — КС-грамматика. Вывод $S^* \Rightarrow \alpha$, где $\alpha \in V^*$, $\alpha \neq \varepsilon$ существует дерево вывода в грамматика G с результатом n

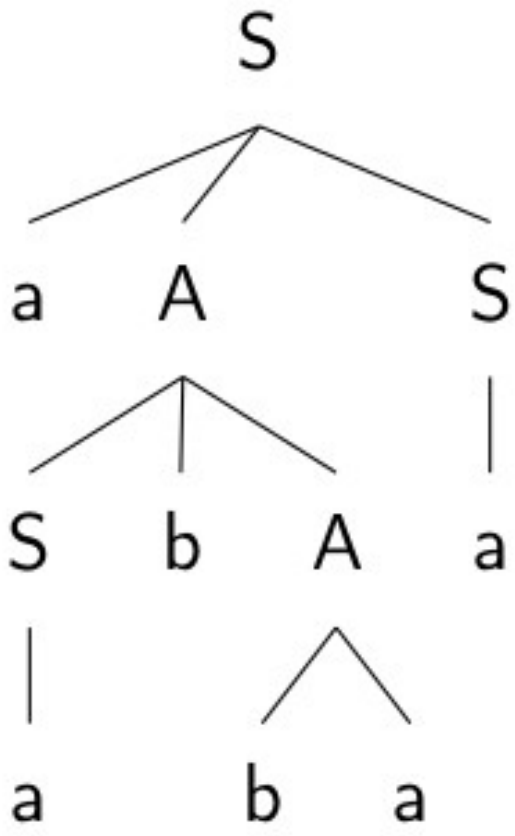


Рисунок 3. Пример дерева вывода

2. Конечные автоматы

2.1. Теоретико-множественное доказательство невозможности описания языков

- *Конечное описание* — предложение над некоторым алфавитом (подразумеваемая интерпретация которого связывает его с описываемым языком)
- Любой язык является не более, чем счётным; соответсвенно, существует не более, чем счетное множество конечных описаний
- Множество всех языков над данным алфавитом не является счетным, т.к. множество всех подмножеств счетного подмножеств более, чем счетно.

Итого, конечных описаний меньше, чем языков, соответсвенно не для всех бесконечных языков существует конечное описание

2.2. Конечные автоматы

Примеры

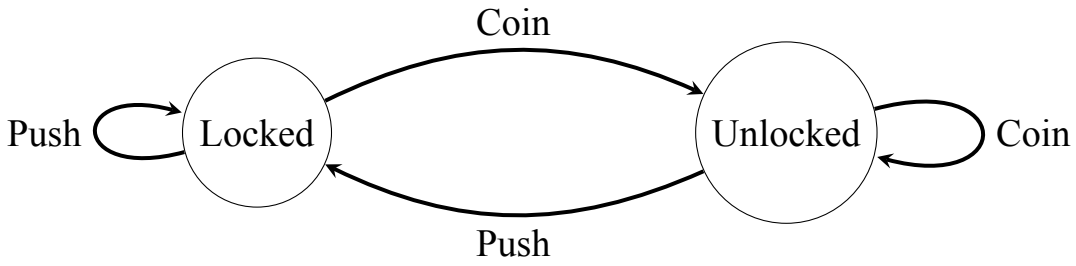


Рисунок 4. Турникет

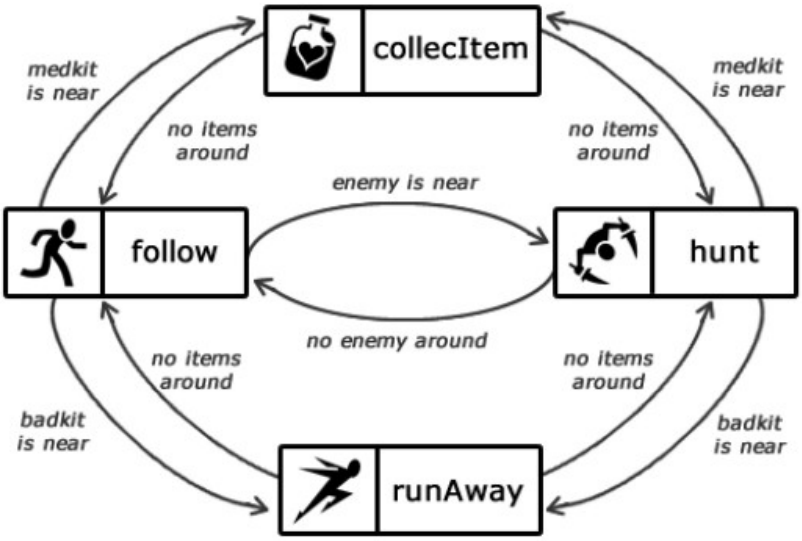


Рисунок 5. Бот

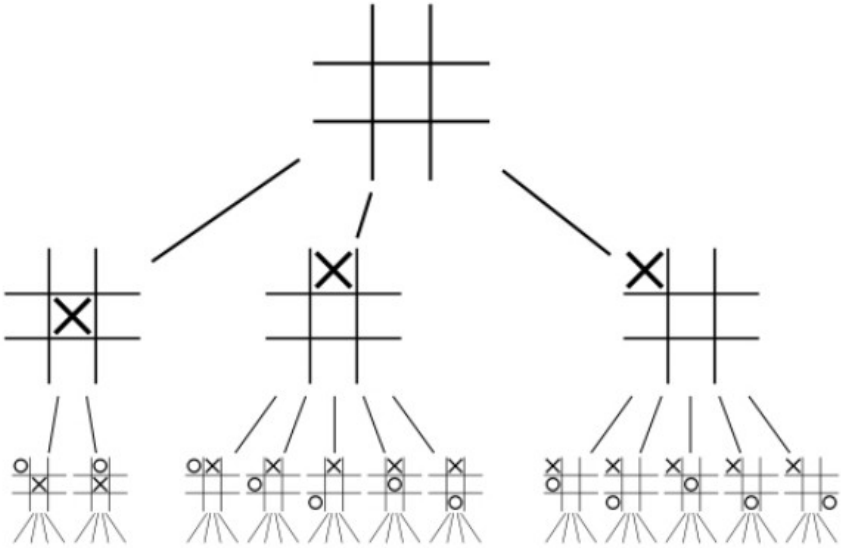


Рисунок 6. Крестики-нолики

2.2.1. Определение

(Детерминированный) конечный автомат — $(Q, \Sigma, \delta, q_0, F)$

- $Q \neq 0$ — конечное множество состояний
- Σ — Конечный входной алфавит
- δ — отображение типа $Q \times \Sigma \to Q$
  $\delta(q_i, x) = y$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

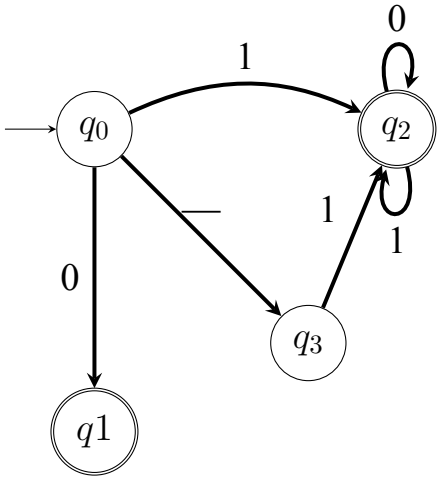
Конечный автомат называется **полным**, если существует переход из каждого состояния по каждому символу алфавита. Если автомат неполный, добавляют *дьявольскую верши-*

ну

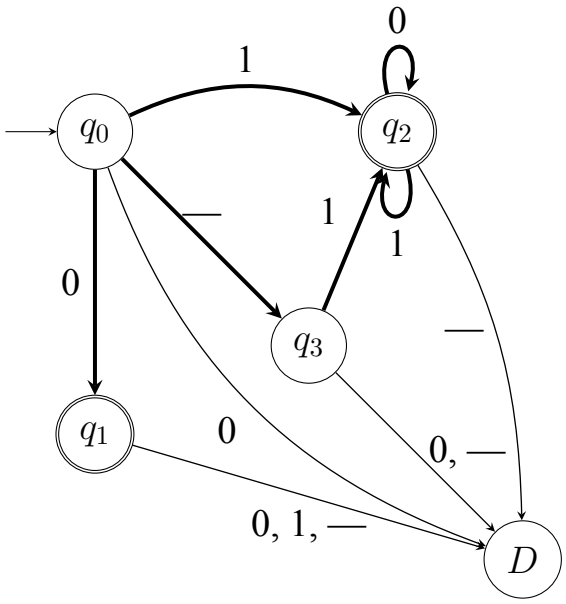
Пример 2.1. Дьявольская вершина

$$Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{0, 1, -\}, q_0 = q_0, F = \{q_1, q_2\}$$

$$\delta(q_0, 0) = q_1$$
$$\delta(q_0, 1) = q_2$$
$$\delta(q_0, -) = q_3$$
$$\delta(q_2, 0) = q_2$$
$$\delta(q_2, 1) = q_2$$
$$\delta(q_3, 1) = q_2$$



(a) Автомат



(b) Автомат с дьявольской вершиной

2.2.2. Путь в конечном автомате

Путь — кортеж $\langle q_0, e_1, q_1, \ldots, e_n, q_n \rangle$

- $n \geq 0$
- $\forall i : e_i = \langle q_{i-1}, w_i, q_i \rangle$, где $\delta(q_{i-1}, w_i) = q_i$
- q_0 — *Начало* пути
- q_n — *Конец* путь
- $w_1, w_2, \ldots w_n$ — *Метка* пути
- n — *Длина* пути

Путь *успешен* , если q_0 — начальное состояние, а $q_n \in F$

Состояние q *достижимо* из состояния p, если существует путь из состояния p в состояние q

Такт работы КА

- *Конфигурация (Мгновенное описание)* КА — $\langle q, \omega \rangle$, где $q \in Q, \omega \in \Sigma^*$
- *Такт работы* — бинарное отношение $\vdash$: если $\delta(p, x) = q$ и $\omega \in \Sigma^*$, то $\langle p, x\omega \rangle \vdash \langle q, \omega \rangle$
- $\vdash^*$ — рефлексивное, транзитивное замыкание $\vdash$

2.2.3. Распознавание слова конечным автоматом

Цепочка ω *распознается* КА, если $\exists$ успешный путь с меткой ω.

Язык, распознаваемый конечным автоматом — $\{\omega \in \Sigma^* | \exists p$ — успешный путь с меткой $\omega\}$

Теорема. Рассмотрим конечный автомат $M = (Q, \Sigma, \delta, q_o, F)$.
Слово $\omega \in \Sigma^*$ принадлежит языку $\Leftrightarrow \exists q \in F : \langle q_0, \omega \rangle \vdash^* \langle q, \varepsilon \rangle$

Обобщение функции перехода:

- $\delta'(q, \varepsilon) = q$
- $\delta'(q, x\alpha) = \delta'(\delta'(q, x), \alpha)$, где $x \in \Sigma^*, \alpha \in \Sigma$

Теорема. Цепочка ω распознается КА $\langle Q, \Sigma, \delta, q_o, F \rangle \Leftrightarrow \exists p \in F : \delta'(q, \omega) = p$

Свойство конкатенации строк

Теорема. $\langle q_1, \alpha \rangle \vdash^* \langle q_2, \varepsilon \rangle , \langle q_2, \beta \rangle \vdash^* \langle q_3, \varepsilon \rangle \Rightarrow \langle q_1, \alpha\beta \rangle \vdash^* \langle q_3, \varepsilon \rangle$

2.3. Эквивалетность конечных автоматов

Конечные автоматы A_1, A_2 *эквивалетны* , если распознают один и тот же язык.

Проверка на эквивалентность — одновременный обход в ширину двух автоматов. Каждый переход должен приводить в терминальне или нетерминальные вершины в обоих

автоматах соответственно

Минимальный конечный автомат — автомат, имеющий наименьшее число состояний, распознающий тот же язык, что и данный.

2.3.1. Классы эквивалетности

Отношение эквивалетности — рефлексивное, симметричное, транзитивное отношение

- xRx
- $xRy \iff yRx$
- $xRy, yRz \Rightarrow xRz$

Теорема. $\forall R$ — отношение эквивалентности на множестве S
Можно разбить S на k непересекающихся подмножеств подмножеств $I_1, \ldots, I_k$, т.ч. $aRb \Leftrightarrow a, b \in I_j$

Множества $I_1 \ldots I_k$ называются *классами эквивалетности*

2.3.2. Эквивалетные состояния

- $\omega \in \Sigma^*$ *различает* состояния q_i и q_j, если $\delta'(q_i, \omega) = t_1, \delta'(q_j, \omega) = t_2 \Rightarrow (t_1 \notin F \Leftrightarrow t_2 \in F)$
- q_i, q_j *эквиваленты* , $(q_i\ q_j)$, если $\forall \omega \in \Sigma^*$: $\delta'(q_i, \omega) = t_1, \delta'(q_j, \omega) = t_2 \Rightarrow (t_1 \in F \Leftrightarrow t_2 \in F)$

Лемма. $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, $p_1, p_2, q_1, q_2 \in Q, q_i = \delta(p_i, c)$. $\omega \in \Sigma^*$ различается q_1 и q_2. Тогда $c\omega$ различает p_1 и p_2.

Доказательство. $\delta'(p_i, c\omega) = \delta'(\delta'(p_i, c), \omega) = \delta'(q_i, \omega) = t_i$

2.3.3. Алгоритм минимизации

Классы эквивалетности разбиваются на новые состояния

- Q — очередь
- `marked` — таблица размером $n \times n$ (n — количество состояний КА)

Помечаем в таблице пары неэквивалетных состояний и кладем их в очередь

1. Если автомат не полный — дополнить дьявольской вершиной

2. Строим отображение δ^{-1} — обратные ребра
3. Находим все достижимые из стартового состояния
4. Добавляем в Q и отмечаем в `marked` пары состояний, различимые ε.
5. Основной цикл. Можем пометить пару (u, v), если $\exists c \in \Sigma : (\delta(u, c), \delta(v, c))$. Для этого, пока $Q \neq \emptyset$:
   - Извлекаем (u, v) из Q
   - $\forall \in \Sigma$ перебираем $(\delta^{-1}(u, c), \delta^{-1}(v, c))$ — если пара не помечена, помечаем и кладем в очередь
6. В момент опустошения Q непомеченные пары являются эквивалетными
7. За проход по таблице выделяем классы эквивалентности
8. За проход по таблице формируем новые состояния и переходы

- *Стартовое состояние* — класс эквивалетности,которому принадлежит стартовое состояние исходного КА
- *Конечные состояния* — классы эквивалетности, которым принадлежат конечные состояния исходного КА

2.4. Недетерминированный КА

- $Q \neq 0$ — конечное множество состояний
- Σ — Конечный входной алфавит
- δ — отображение типа $Q \times \Sigma \to 2^Q$
  $\delta(q_i, x) = \{q_{j_0} \dots q_{j_k}\}$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

Отличается отображение. Из состояния по символу происходит переход в некоторое множество состояний. В графическом виде это значит, что из одной вершины по одному символу выходит одна или несколько стрелок.

Такт работы связывает две конфигурации.

- *Конфигурация (Мгновенное описание)* — $\langle q, \omega \rangle$, где $q \in Q, \omega \in \Sigma^*$
- *Такт работы* — бинарное отношение $\vdash$: если $q \in \delta(p, x)$ и $\omega \in \Sigma^*$, то $\langle p, x\omega \rangle \vdash \langle q, \omega \rangle$
- $\vdash^*$ — рефлексивное, транзитивное замыкание $\vdash$
- НКА допускает слово α, если $\exists t \in F : \langle s, \alpha \rangle \vdash^* \langle t, \varepsilon \rangle$

- *Язык НКА* $L(a) = \{\omega \in \Sigma^* | \exists t \in F : \langle s, \omega \rangle \vdash^* \langle t, \varepsilon \rangle\}$
- ДКА — частный случай НКА

Алгоритм Томпсона

- Q — очередь. Помещаем туда $\{q_0\}$.
- Пока $Q \neq \emptyset$:
  - $q \leftarrow$ Q.pop()
  - $q' = \{t = \delta(s, c) | s \in q, c \in \Sigma\}$. Если $q' \notin Q$, добавить его в очередь.
    Каждое такое множество — новая вершина ДКА; добавляются переходы по соответсвующим символам
  - Если во множестве есть хотябы одна вершина, являющаяся терминальное в НКА, то в ДКА соответствующая вершина будет конечной
- Результат: $\{\Sigma, Q_d, q_{d_0} \in Q, F_d \subset Q_d, \delta_d : Q_d \times \Sigma \to Q_d\}$
  - $Q_d = \{q_d | q_d \subseteq 2^Q\}$
  - $q_{d_0} = \{q_0\}$
  - $F_d = \{q \in Q_d | \exists p \in F : p \in q\}$
  - $\delta_d(q, c) = \{\delta(a, c) | a \in q\}$

ДКА строить сложнее, но их легче использовать — если длина цепочки n, то можно проверить, принадлежит ли она языку за $o(n)$.

Теорема. ДКА и НКА распознают один и тот же класс языков

Доказательство.
$\Rightarrow$ — очевидно
$\Leftarrow$ — Рассмотрим произвольный НКА и покажем, что алгоритм Томпсона строит по нему эквивалетный ДКА.

$$\forall q in q_d, \forall c \in \Sigma^*, \forall p \in \delta(q, c) : p \in \delta_d(q_d, c)$$

Рассмотрим $\langle q_0, w_1 w_2 \dots w_m \rangle \vdash \langle u_1, w_2 \dots w_m \rangle \vdash^* \langle u_m, \varepsilon \rangle, u_m \in F$

$$\forall i : u_i \in u_{d_i}, \text{где} (q_{d_0}, w_1 w_2 \dots w_m) \vdash (u_{d_1}, w_2 \dots w_m) \vdash^* (u_{d_m}, \varepsilon) \Rightarrow u_m \in u_{d_m}$$