Теория языков программирования и методы трансляции

Романенко Владимир Васильевич,
к.т.н., доцент каф. АСУ ТУСУР

Введение

Язык L_1 → Язык L_2

Трансляция программы

Введение

1. Лексический анализ
2. Синтаксический анализ
3. Генерация кода
4. Оптимизация кода

Организация трансляции программы

Введение

Используемые методы:

- теория языков;
- теория перевода;
- методы синтаксического анализа.

База:

- математическая подготовка (булева алгебра, дискретная математика);
- теория множеств;
- Знание ОО-языка высокого уровня.

Предварительные математические сведения

Базовые понятия:

- атомы;
- множества;
- элементы множеств;
- предикаты;
- операции над множествами;
- отношения на множествах;
- замыкание отношений.

Предварительные математические сведения

Множества цепочек

- *Алфавит* Σ – множество символов, из которых состоят *цепочки символов* (*предложения*) анализируемого языка.
- *Символ* – элемент алфавита ($a \in \Sigma$).

Примеры:

- Двоичный алфавит $\Sigma = \{0, 1\}$.
- Алфавит русского языка Σ = {а б в … я А Б В … Я 0 1 2 … 9 . , ! ? …} или {а-я А-Я 0-9 . , ! ? …}

Множества цепочек

Формально цепочки в алфавите Σ определяются следующим образом:

1) e – цепочка в Σ;

2) если α цепочка в Σ и $a \in \Sigma$, то αa – цепочка в Σ;

3) β – цепочка в Σ тогда и только тогда, когда она является таковой в силу 1) и 2).

Если цепочку из i символов a обозначить как a^i, тогда $a^0 = e$ – пустая цепочка.

Множества цепочек

- *Цепочка* α – последовательность из 0 и более символов алфавита Σ: $\alpha \in \Sigma^*$.
- *Пустая цепочка* – цепочка, состоящая из 0 символов ($\alpha = e$).
- Цепочка из 1 и более символов – $\alpha \in \Sigma^+$.

Пример: пусть $\Sigma = \{0, 1\}$, тогда:

- $\alpha = 010 \in \Sigma^+$ и $\alpha = 010 \in \Sigma^*$;
- $\alpha = e \notin \Sigma^+$ и $\alpha = e \in \Sigma^*$;
- $\alpha = e1e1e = 11 \in \Sigma^+$ и $\alpha = e1e1e = 11 \in \Sigma^*$.

Операции над цепочками

- $\alpha\beta$ – конкатенация (сцепление) цепочек α и β. Например, если $\alpha = ab$, $\beta = cd$, то $\alpha\beta = abcd$. При этом пустая цепочка играет роль «единицы»: $\alpha e = e\alpha = \alpha$. В сцепленной цепочке $\alpha\beta\gamma$ цепочка α называется префиксом (если $\beta\gamma \neq e$, то собственным), цепочка β называется подцепочкой, а цепочка γ – суффиксом (если $\alpha\beta \neq e$, то собственным) цепочки $\alpha\beta\gamma$.
- $|\alpha|$ – длина цепочки. Например, $|abcd| = 4$, $|e| = 0$.

Языки

- *Язык L* – множество правильных цепочек (предложений) в алфавите Σ: $L \subseteq \Sigma^*$ или $L \subseteq \Sigma^+$.

Примеры:

- Двоичные числа: $\Sigma = \{0, 1\}$,
  $L = \Sigma^+ = \{0, 1, 00, 01, 10, 11, 001, 010, 011, 100, \ldots\}$.
- Двоичные числа без незначащих нулей: $\Sigma = \{0, 1\}$,
  $L \subset \Sigma^+ = \{0, 1, 10, 11, 100, 101, 110, 111, 1000, \ldots\}$.

Языки

- *Язык L* – множество правильных цепочек (предложений) в алфавите Σ: $L \subseteq \Sigma^*$ или $L \subseteq \Sigma^+$.

Примеры:

- Русский язык: Σ = {а-я А-Я 0-9 . , ! ? …},
  $L \subset \Sigma^+$ = {мама, папа, я, …}.
- Язык C++: Σ = {_ a-z A-Z 0-9 , ; : # < > ( ) …},
  $L \subset \Sigma^*$ = {*e*, void main() {}, #include <vector>, …}.

Соглашения

- Прописные буквы греческого алфавита – алфавиты.
- Прописные буквы латинского алфавита – множества, языки, состояния анализаторов (в т.ч. нетерминалы грамматик).
- Строчные буквы греческого алфавита – цепочки символов.
- Строчные буквы латинского алфавита – отдельные символы.

Задачи, решаемые методами синтаксического анализа:

1. Получение всех правильных цепочек (предложений) языка, т.е. построение *дерева вывода*.
2. Проверка принадлежности цепочки (предложения) языку ($\alpha \in L$ или $\alpha \notin L$).

Введение в компиляцию

Как описать язык L?

- Если язык состоит из конечного числа цепочек (предложений), то можно просто составить список всех цепочек (предложений).

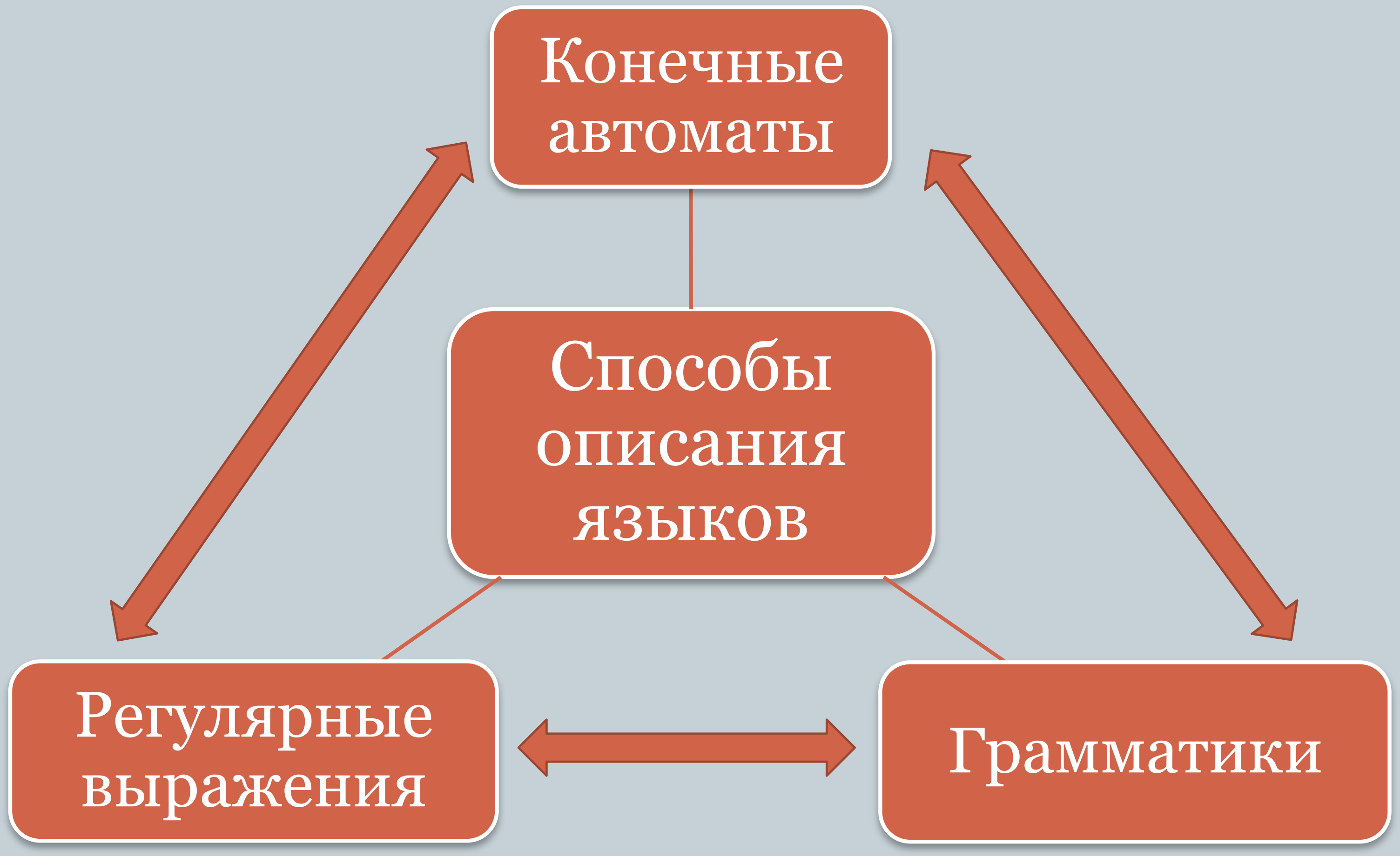
Пример: язык булевых констант L = {false, true}.

Пример: двоичные числа от 0 до 255 (байт).

- Если число цепочек (предложений) бесконечно, то требуются специальные средства описания языков.

Введение в компиляцию

Введение в компиляцию



Конечные автоматы

Классификация КА

- Конечный автомат (КА)
  - Недетерминированный конечный автомат (НКА)
  - Детерминированный конечный автомат (ДКА)
    - ДКА с магазинной памятью (ДМПА)

Формальное описание КА

Конечный автомат (КА):

$$M = (Q, \Sigma, \delta, q_0, F),$$

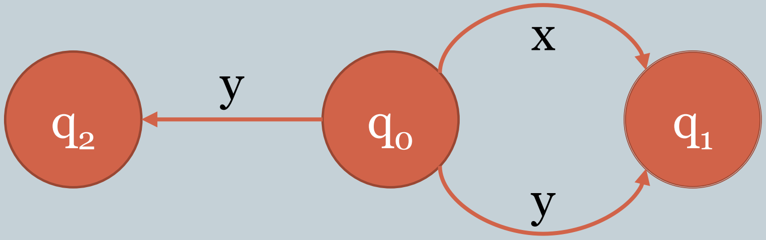
где

- Q – конечное множество состояний;
- Σ – конечное множество входных символов (алфавит);
- δ – функция переходов, отображение множества $Q\times\Sigma$ во множество $\boldsymbol{P}(Q)$;
- $q_0 \in Q$ – начальное состояние;
- $F \subseteq Q$ – множество заключительных состояний.

Формальное описание КА

Способы записи функции переходов:

- В виде отображения:
  $\delta(q_0, x) = \{q_1\}, \delta(q_0, y) = \{q_1, q_2\}, \dots$
- В виде множества:
  $\delta = \{((q_0, x), \{q_1\}), ((q_0, y), \{q_1, q_2\}), \dots\}$
- В виде графа:



Формальное описание КА

Способы записи функции переходов:

- В виде таблицы:

	x	**y**
q_0	$\{q_1\}$	$\{q_1, q_2\}$
q_1	$\{\}$	$\{\}$
q_2	$\{\}$	$\{\}$

В детерминированном автомате $\#\delta(q, a) \le 1$.
Данный автомат недетерминирован, т.к.

$$\#\delta(q_0, y) = \#\{q_1, q_2\} = 2.$$

Формальное описание ДКА

Детерминированный конечный автомат (ДКА):

$$M = (Q, \Sigma, \delta, q_0, F),$$

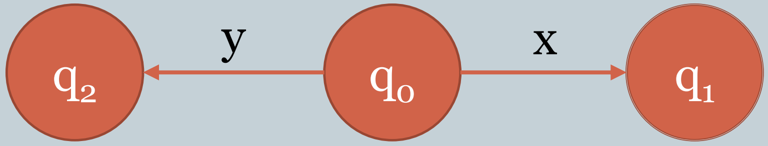
где

- Q – конечное множество состояний;
- Σ – конечное множество входных символов (алфавит);
- δ – функция переходов, отображение множества $Q \times \Sigma$ во множество Q;
- $q_0 \in Q$ – начальное состояние;
- $F \subseteq Q$ – множество заключительных состояний.

Формальное описание ДКА

Способы записи функции переходов:

- В виде отображения:
  $\delta(q_0, x) = q_1, \delta(q_0, y) = q_2, \dots$
- В виде множества:
  $\delta = \{((q_0, x), q_1), ((q_0, y), q_2), \dots\}$
- В виде графа:



Формальное описание ДКА

Способы записи функции переходов:

- В виде таблицы:

	x	y
q_0	q_1	q_2
q_1	ERROR	ERROR
q_2	ERROR	ERROR

ERROR – специальный символ ошибки синтаксического анализа (в функции переходов его можно не указывать).

Формальное описание ДКА

Сокращенный вариант записи:

$$M = (Q, \Sigma, \delta, q_0),$$

где

- Q – конечное множество состояний;
- Σ – конечное множество входных символов (алфавит);
- δ – функция переходов, отображение множества $Q\times(\Sigma\cup\{\perp\})$ во множество Q;
- $q_0 \in Q$ – начальное состояние.

Формальное описание ДКА

Новые обозначения:

- ⊥ – маркер конца цепочки (должен присутствовать в конце каждой анализируемой цепочки);
- HALT – специальный символ, означает успешное завершение разбора.

Формальное описание ДКА

Таблица переходов ДКА:

	a_1	a_2	…	$\perp$
q_0	$\delta(q_0, a_1)$	$\delta(q_0, a_2)$	…	$\delta(q_0, \perp)$
q_1	$\delta(q_1, a_1)$	$\delta(q_1, a_2)$	…	$\delta(q_1, \perp)$
q_2	$\delta(q_2, a_1)$	$\delta(q_2, a_2)$	…	$\delta(q_2, \perp)$
…	…	…	…	…

$\delta(q, a)$:

- q';
- $HALT\,(a = \perp, q \in F)$;
- $ERROR$.

Пример описания ДКА

Пусть язык описывает двоичные числа без незначащих нулей. Тогда:

- $Q = \{q_0, q_1, q_2\}$;
- $\Sigma = \{0, 1\}$;
- $\delta = \{((q_0, 0), q_1), ((q_0, 1), q_2), ((q_1, 0), ERROR), ((q_1, 1), ERROR), ((q_2, 0), q_2), ((q_2, 1), q_2)\}$;
- $q_0 = q_0$.
- $F = \{q_1, q_2\}$.

Т.е. $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{((q_0, 0), q_1), ((q_0, 1), q_2), ((q_2, 0), q_2), ((q_2, 1), q_2)\}, q_0, \{q_1, q_2\})$

Пример описания ДКА

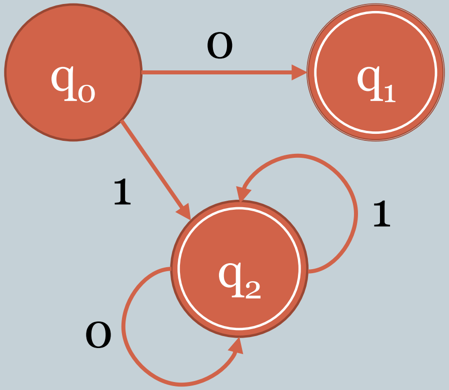
Или:

- $Q = \{q_0, q_1, q_2\}$;
- $\Sigma = \{0, 1\}$;
- $\delta = \{((q_0, 0), q_1), ((q_0, 1), q_2), ((q_0, \perp), ERROR), ((q_1, 0), ERROR), ((q_1, 1), ERROR), ((q_1, \perp), HALT), ((q_2, 0), q_2), ((q_2, 1), q_2), ((q_2, \perp), HALT)\}$;
- $q_0 = q_0$.

Т.е. $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{((q_0, 0), q_1), ((q_0, 1), q_2), ((q_1, \perp), HALT), ((q_2, 0), q_2), ((q_2, 1), q_2), ((q_2, \perp), HALT), q_0)$

Пример описания ДКА

Функция переходов в виде графа:



Функция переходов в виде таблицы:

	0	**1**	**$\perp$**
q_0	q_1	q_2	ERROR
q_1	ERROR	ERROR	HALT
q_2	q_2	q_2	HALT

Пример описания ДКА

Состояниям можно давать любые имена:

	0	1	⊥
start	zero	non-zero	ERROR
zero	ERROR	ERROR	HALT
non-zero	non-zero	non-zero	HALT

Тогда:

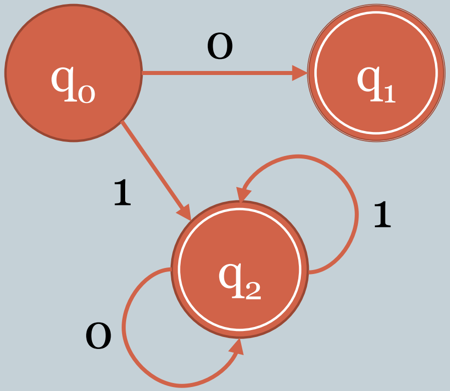
- Q = {start, zero, non-zero};
- q_0 = start;
- и т.д.

Проверка ДКА

Как проверить, что ДКА описан правильно?

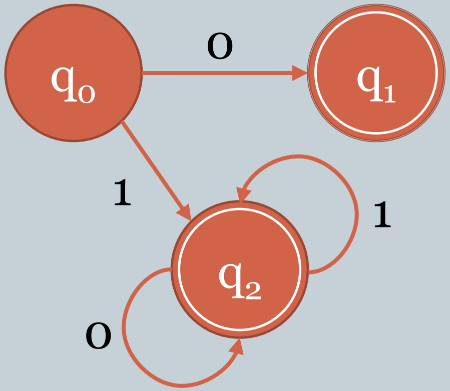
1. Построить дерево вывода. Если дерево содержит все правильные цепочки языка, и не содержит неправильные цепочки, то автомат описан правильно.
2. Осуществить запуск ДКА. Если для всех правильных цепочек разбор окончится символом HALT, а всех неправильных – символом ERROR, то автомат описан правильно.

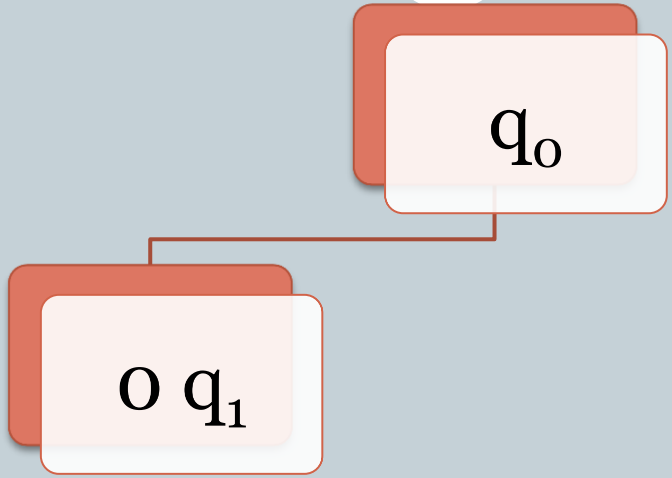
Дерево вывода ДКА



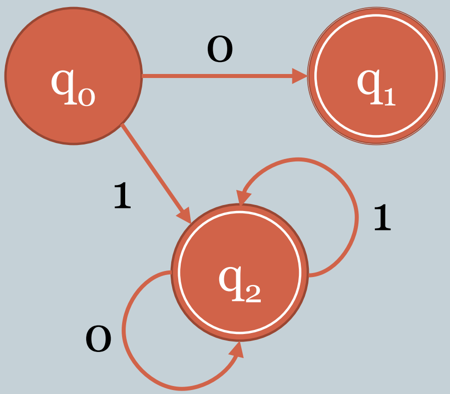
q_0

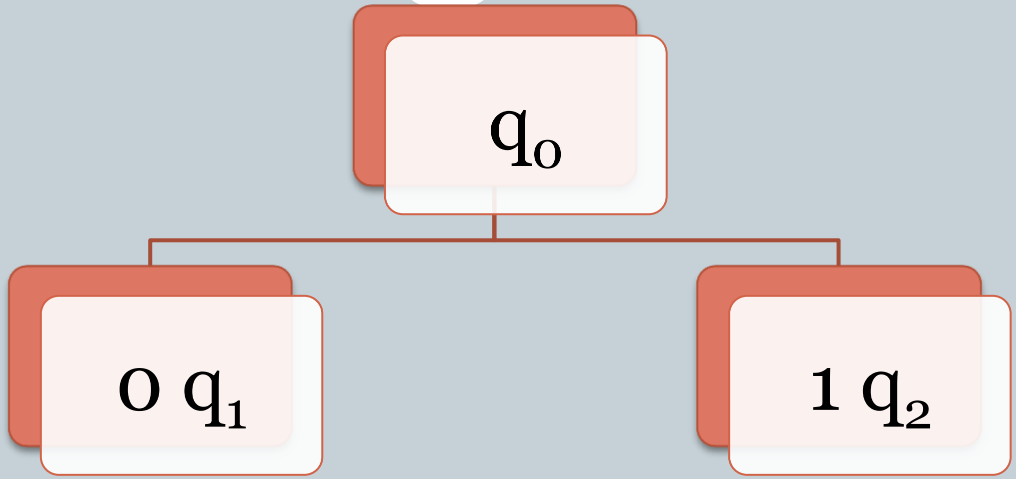
Дерево вывода ДКА



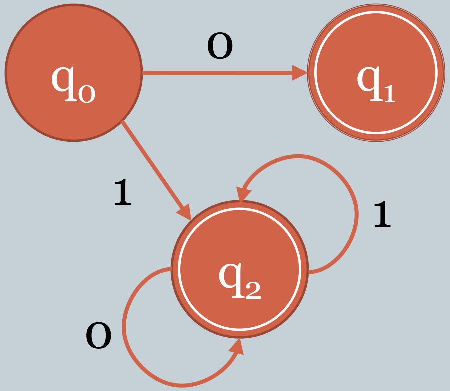


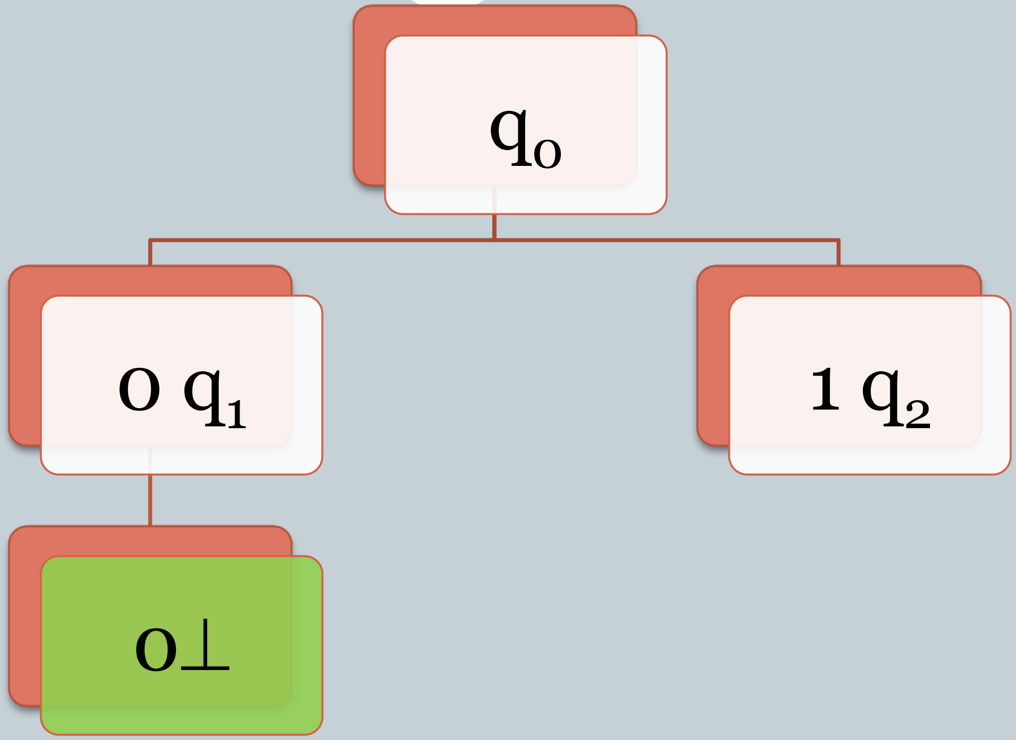
Дерево вывода ДКА



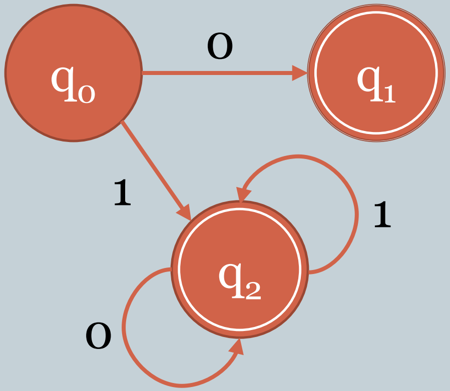


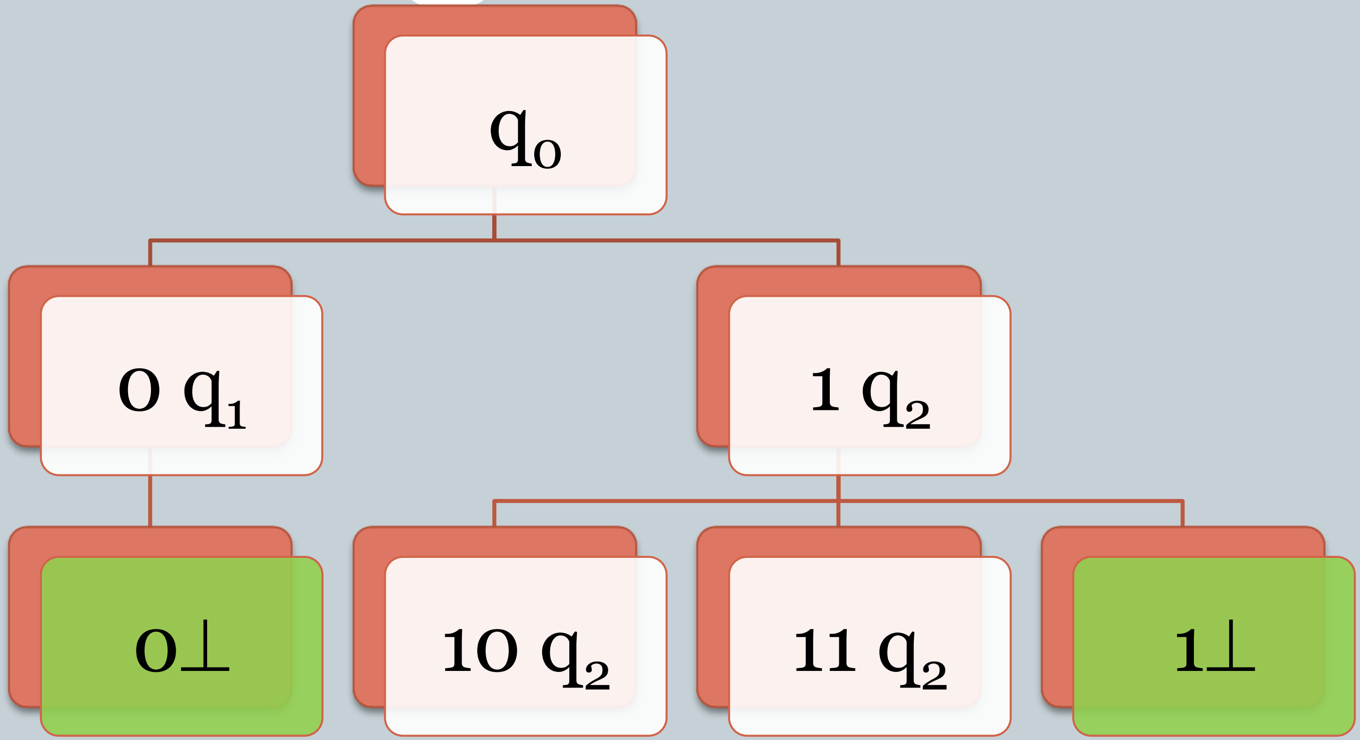
Дерево вывода ДКА



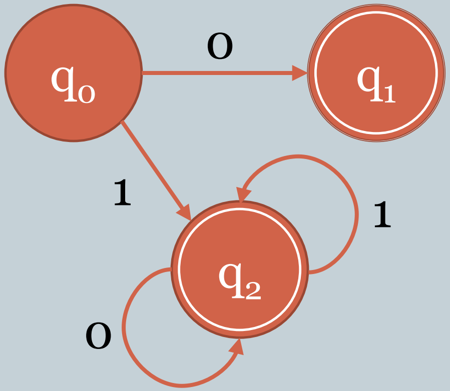


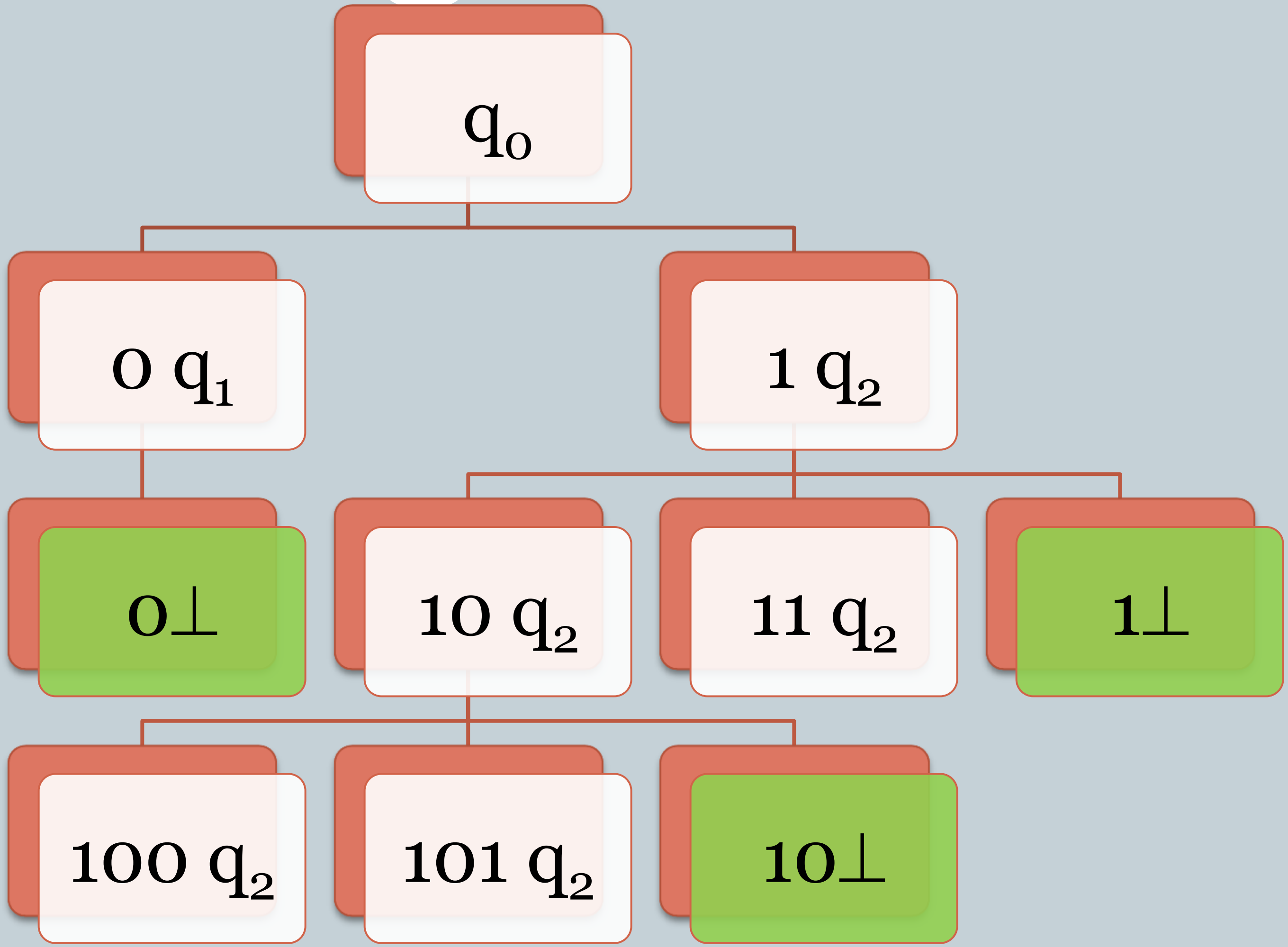
Дерево вывода ДКА



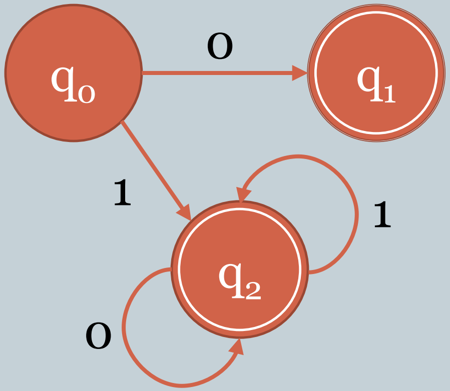


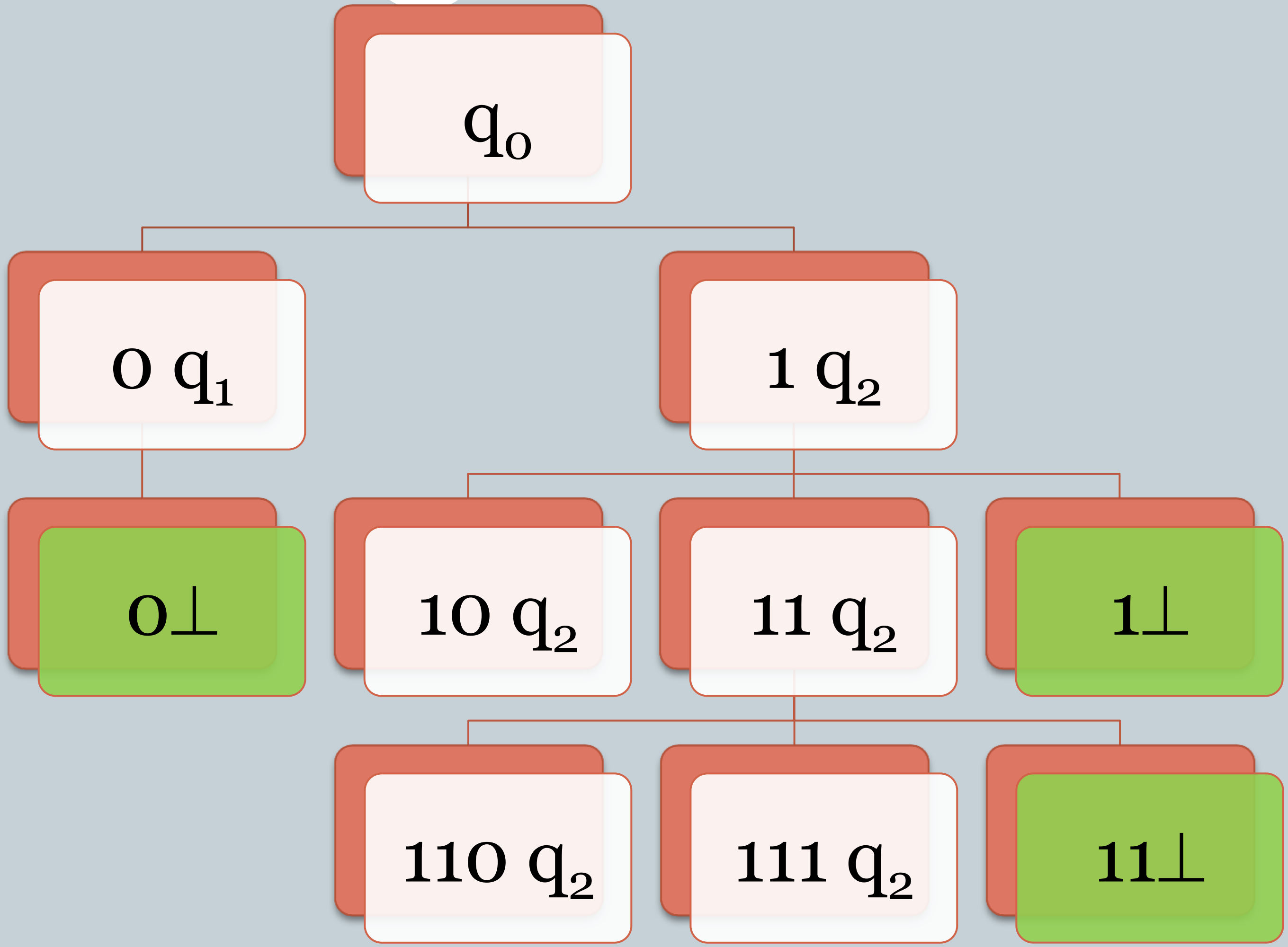
Дерево вывода ДКА





Дерево вывода ДКА





Запуск ДКА

Основные понятия:

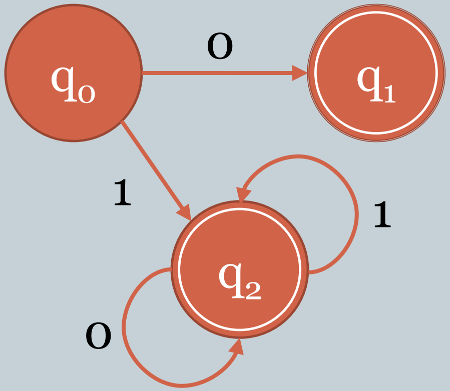
- *конфигурация* ДКА M

$$(q, \alpha) \in Q \times \Sigma^*;$$

- *начальная конфигурация* – (q_0, α), где $\alpha \in \Sigma^*$;
- *заключительная конфигурация* – $(q, \perp)$, где $q \in F$;
- *такт* работы ДКА M при $\delta(q, a) = q'$, где $q, q' \in Q$, $a \in \Sigma \cup \{\perp\}$:

$$(q, a\alpha) \Rightarrow (q', \alpha)$$

Запуск ДКА



- Для правильной цепочки:

$(q_0, «110\perp») \Rightarrow^1 (q_2, «10\perp»)$
$\Rightarrow^2 (q_2, «0\perp»)$
$\Rightarrow^3 (q_2, «\perp»)$
$\Rightarrow^4$ HALT

- Для неправильной цепочки:

$(q_0, «010\perp») \Rightarrow^1 (q_1, «10\perp»)$
$\Rightarrow^2$ ERROR

Алгоритм работы ДКА

Пусть $\alpha = a_1a_2\ldots a_n\perp$ – входная цепочка. Тогда:

1. Полагаем $q \leftarrow q_0$, $k \leftarrow 1$.

2. Ищем $\delta(q, a)$, где $a = a_k$.

3. Если $\delta(q, a)$ не определена, то ошибка в позиции k. Если значений $\delta(q, a)$ несколько – таблица переходов построена неверно. Если $\delta(q, a) = q'$, то:

   3.1. Переходим в новое состояние $q \leftarrow q'$.

   3.2. Переходим к следующему символу $k \leftarrow k + 1$.

4. Если $\delta(q, a) = HALT$, то разбор успешно завершен.

5. Если $\delta(q, a) = ERROR$, то имеем во входной цепочке синтаксическую ошибку в позиции k.

6. Иначе возврат на шаг 2.

Минимизация ДКА

Цель минимизации – избавиться от лишних состояний, которые дублируют друг друга.

Минимизация ДКА

Цель минимизации – избавиться от лишних состояний, которые дублируют друг друга.

Способ 1. Сравним два состояния q_i и q_j. Поведение автомата в этих двух состояниях идентично, если:

- Функция переходов для этих состояний дает одинаковый результат:

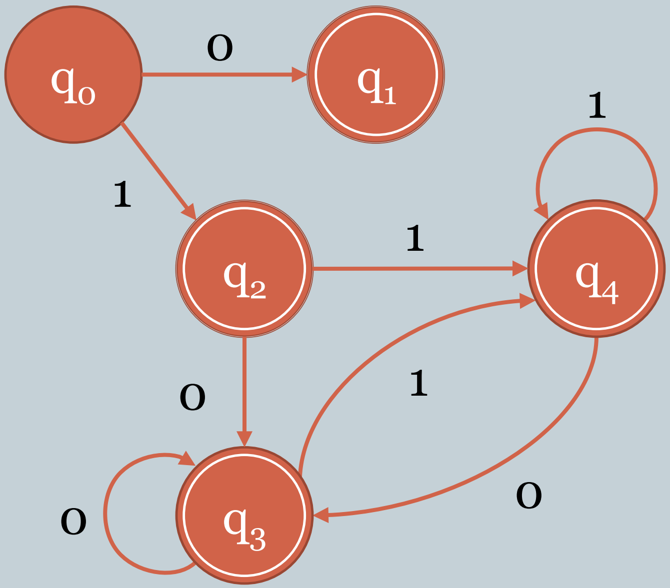
$$\delta(q_i, a) = \delta(q_j, a) \quad \forall a \in \Sigma;$$

- Оба состояния являются конечными, либо оба не являются таковыми:

$$q_i, q_j \in F \text{ или } q_i, q_j \notin F.$$

Минимизация ДКА

Пример 1.



Минимизация ДКА

Пример 1.

	0	1	$\perp$
q_0	q_1	q_2	
q_1			HALT
q_2	q_3	q_4	HALT
q_3	q_3	q_4	HALT
q_4	q_3	q_4	HALT

Минимизация ДКА

Пример 1.

	0	1	⊥
q_0	q_1	q_2	
q_1			HALT
q_2	q_3	q_4	HALT
q_3	q_3	q_4	HALT
q_4	q_3	q_4	HALT

$\Rightarrow q_3 = q_4$

Минимизация ДКА

Пример 1.

	0	1	⊥
q_0	q_1	q_2	
q_1			HALT
q_2	q_3	$\cancel{q_4}$ q_3	HALT
q_3	q_3	$\cancel{q_4}$ q_3	HALT
~~q_4~~	~~q_3~~	~~q_4~~	~~HALT~~

Минимизация ДКА

Пример 1.

	0	1	$\perp$
q_0	q_1	q_2	
q_1			HALT
q_2	q_3	q_3	HALT
q_3	q_3	q_3	HALT

Минимизация ДКА

Пример 1.

	0	1	$\perp$
q_0	q_1	q_2	
q_1			HALT
q_2	q_3	q_3	HALT
q_3	q_3	q_3	HALT

$\Rightarrow q_2 = q_3$

Минимизация ДКА

Пример 1.

	0	**1**	$\perp$
q_0	q_1	q_2	
q_1			HALT
q_2	$\cancel{q_3}$ q_2	$\cancel{q_3}$ q_2	HALT
$\cancel{q_3}$	$\cancel{q_3}$	$\cancel{q_3}$	~~HALT~~

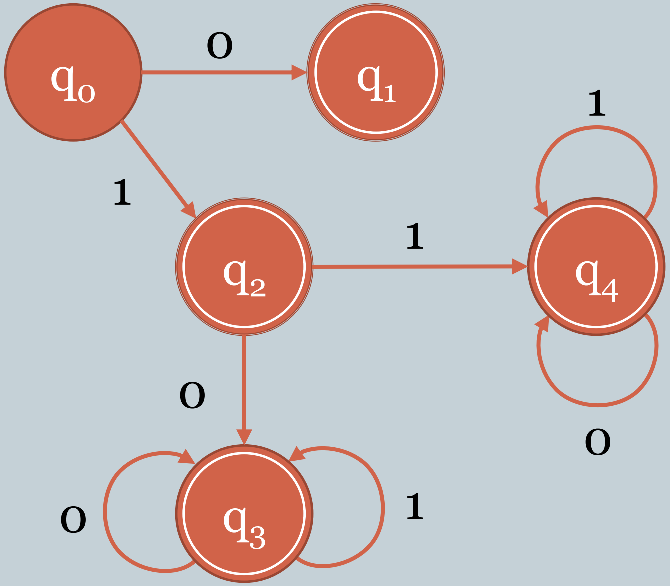
Минимизация ДКА

Пример 1.

	0	**1**	**$\perp$**
q_0	q_1	q_2	
q_1			HALT
q_2	q_2	q_2	HALT

Минимизация ДКА

Пример 2.



Минимизация ДКА

Пример 2.

	0	1	$\perp$
q_0	q_1	q_2	
q_1			HALT
q_2	q_3	q_4	HALT
q_3	q_3	q_3	HALT
q_4	q_4	q_4	HALT

Минимизация ДКА

Способ 2. Сравним два состояния q_i и q_j. Поведение автомата в этих двух состояниях идентично, если:

- Функция переходов для этих состояний дает одинаковый результат:

$$\delta'(q_i, a) = \delta'(q_j, a) \quad \forall a \in \Sigma.$$

Здесь $\delta'(q, a) = q'$, если $q = q_i$ или $q = q_j$, в противном случае $\delta'(q, a) = \delta(q, a)$.

- Оба состояния являются конечными, либо оба не являются таковыми:

$$q_i, q_j \in F \text{ или } q_i, q_j \notin F.$$

Минимизация ДКА

Пример 2.

	0	1	⊥
q_0	q_1	q_2	
q_1			HALT
q_2	q_3	q_4	HALT
q_3	q'	q'	HALT
q_4	q'	q'	HALT

$$\Rightarrow q_3 = q_4$$

Минимизация ДКА

Пример 2.

	0	**1**	**⊥**
q_0	q_1	q_2	
q_1			HALT
q_2	q_3	$\cancel{q_4}$ q_3	HALT
q_3	q_3	q_3	HALT
~~q_4~~	~~q'~~	~~q'~~	~~HALT~~

Определение функции переходов

1. Построить граф переходов, а потом преобразовать его в таблицу переходов.

2. Построение графа начинается с начального состояния q_0. Если начальное состояние может являться также и конечным (т.е. язык допускает пустые предложения), помечаем его двойной границей.

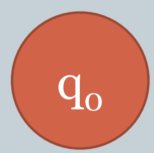
3. Для каждого состояния графа q_i определяем, есть ли из данного состояния такие переходы, которые соответствуют допустимому символу a из входной цепочки, которые пока еще отсутствуют в графе. Если есть, то проверяем, ведет ли данный переход в уже имеющееся состояние q_j. Если да, то добавляем в граф только новый переход $\delta(q_i, a) = q_j$. Если нет, то добавляем в граф новое состояние q_k и переход в него $\delta(q_i, a) = q_k$. Если новое состояние может являться конечным, помечаем его двойной границей.

4. Если в процессе выполнения шага 3 в графе появились новые состояния или переходы, возвращаемся на шаг 3, иначе граф переходов построен.

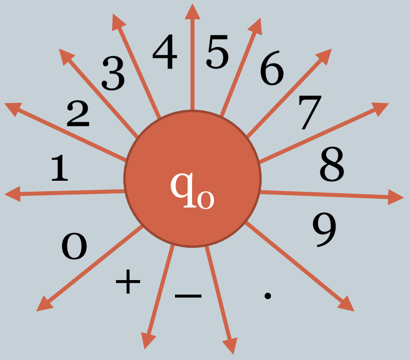
Определение функции переходов

Пример. Рассмотрим язык *L*, описывающий число с фиксированной точкой. Такое число может начинаться со знака «+» или «–», далее следует мантисса числа. Разные языки программирования допускают различные формы записи мантиссы, в общем случае они могут быть следующими: «*N.M*», «*N.*», «*.M*», «*N*», где *N* – целая, а *M* – дробная часть числа. Оба числа *N* и *M* имеют одинаковый формат – это последовательность из одной и более цифр в диапазоне от 0 до 9.

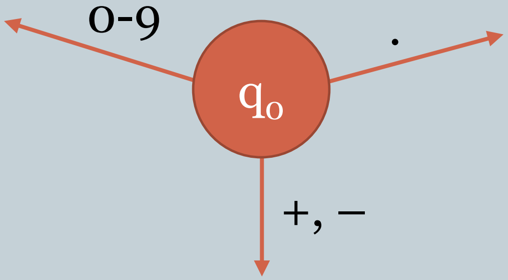
Определение функции переходов



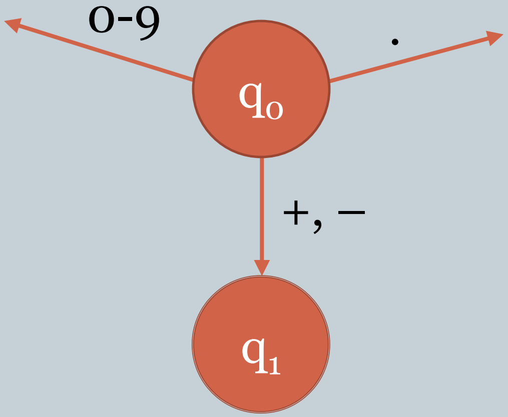
Определение функции переходов



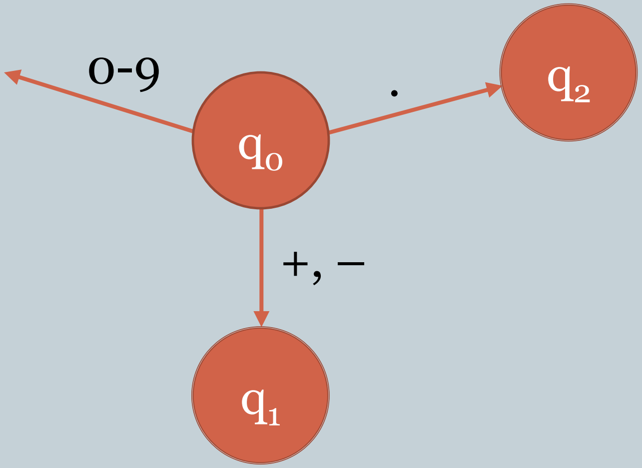
Определение функции переходов



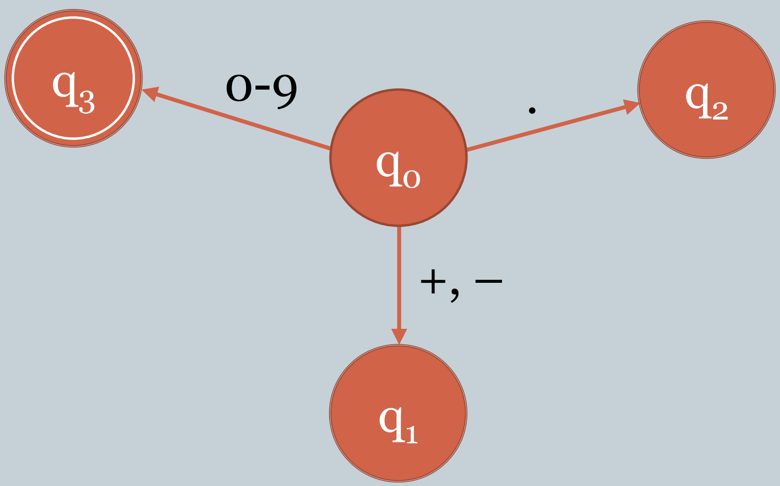
Определение функции переходов



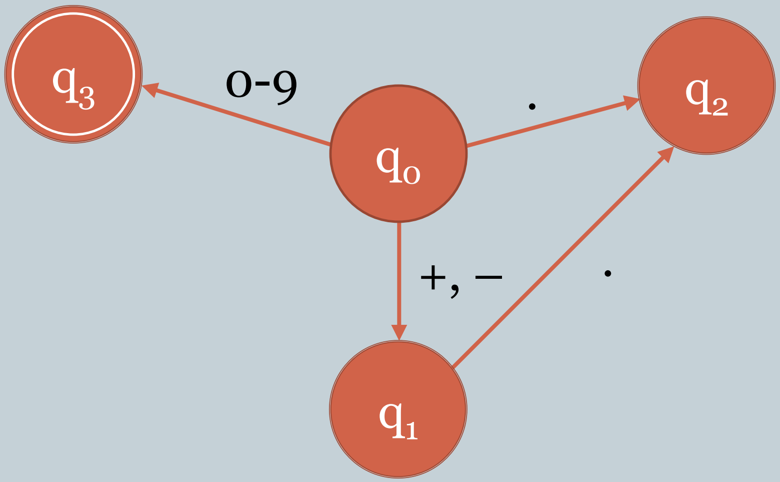
Определение функции переходов



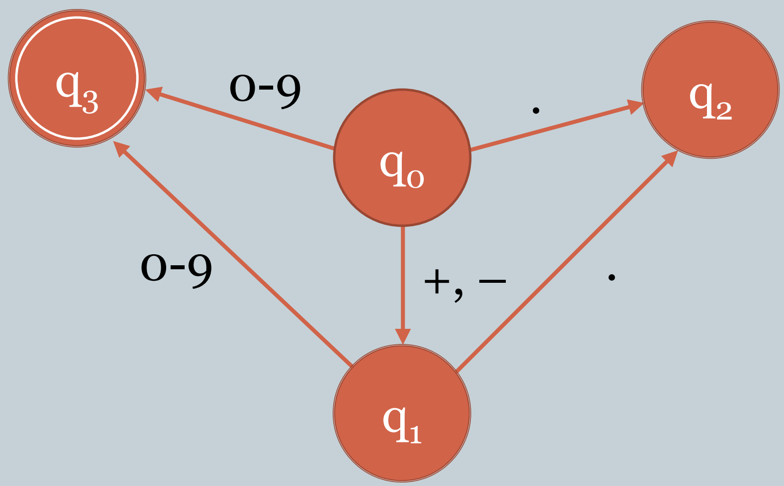
Определение функции переходов



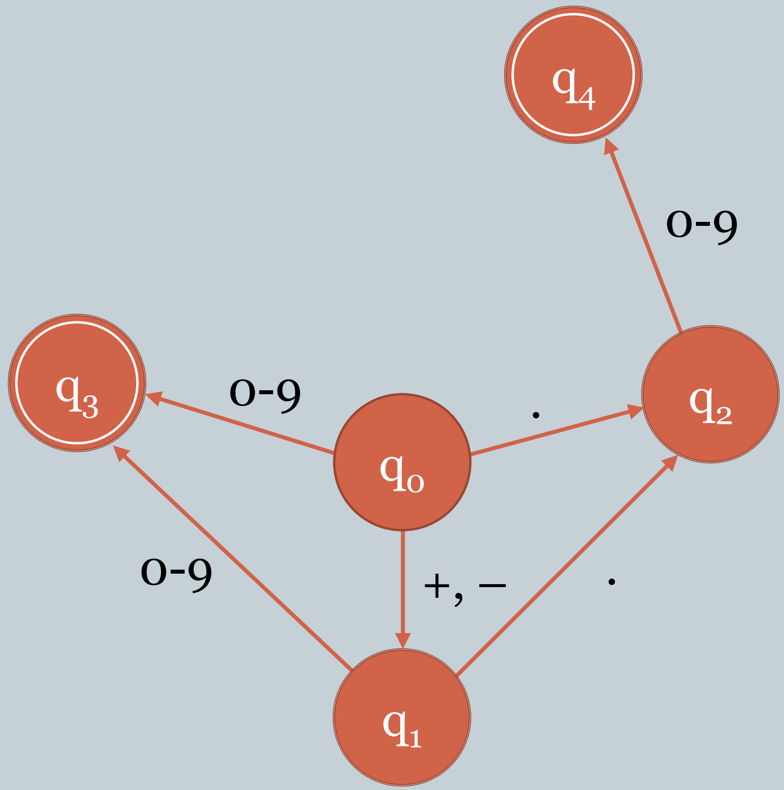
Определение функции переходов



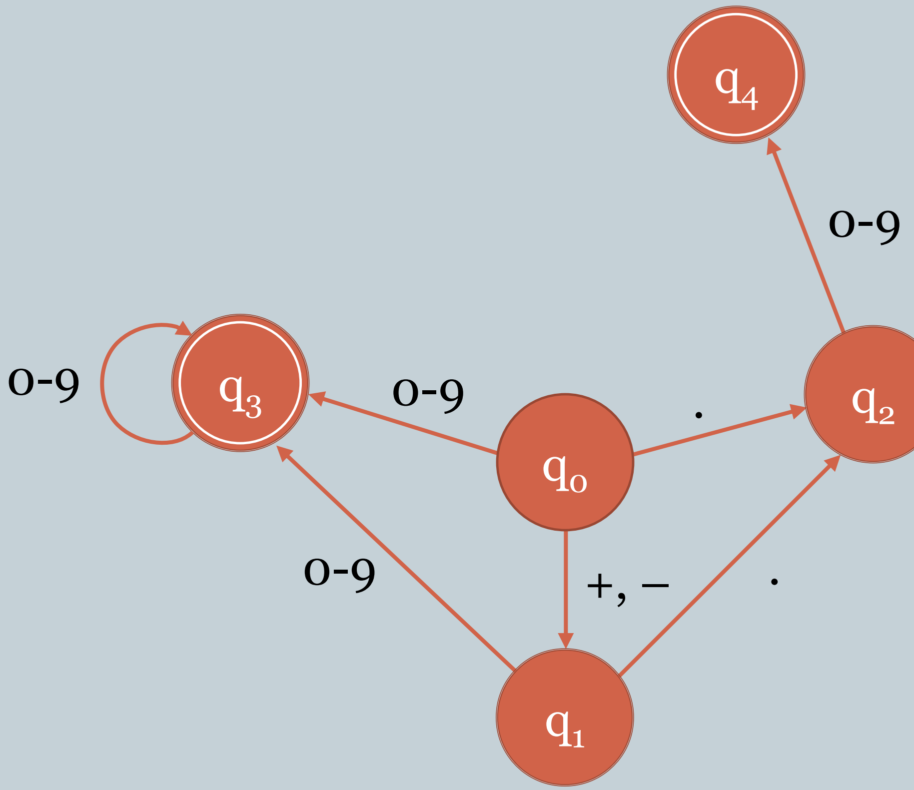
Определение функции переходов



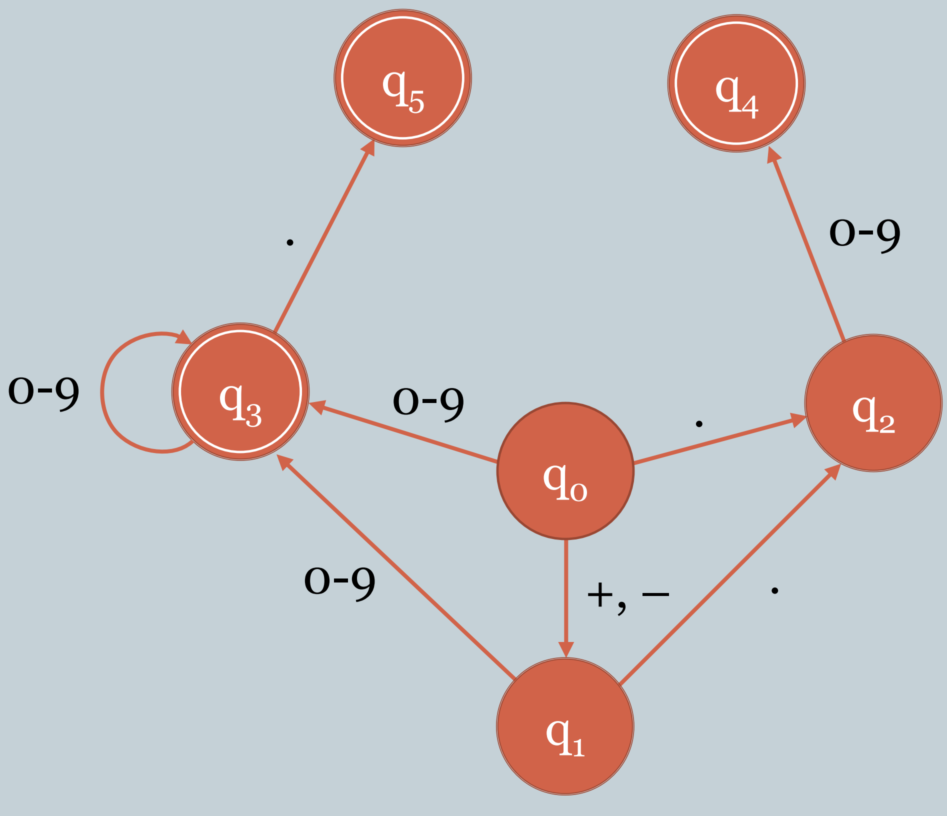
Определение функции переходов



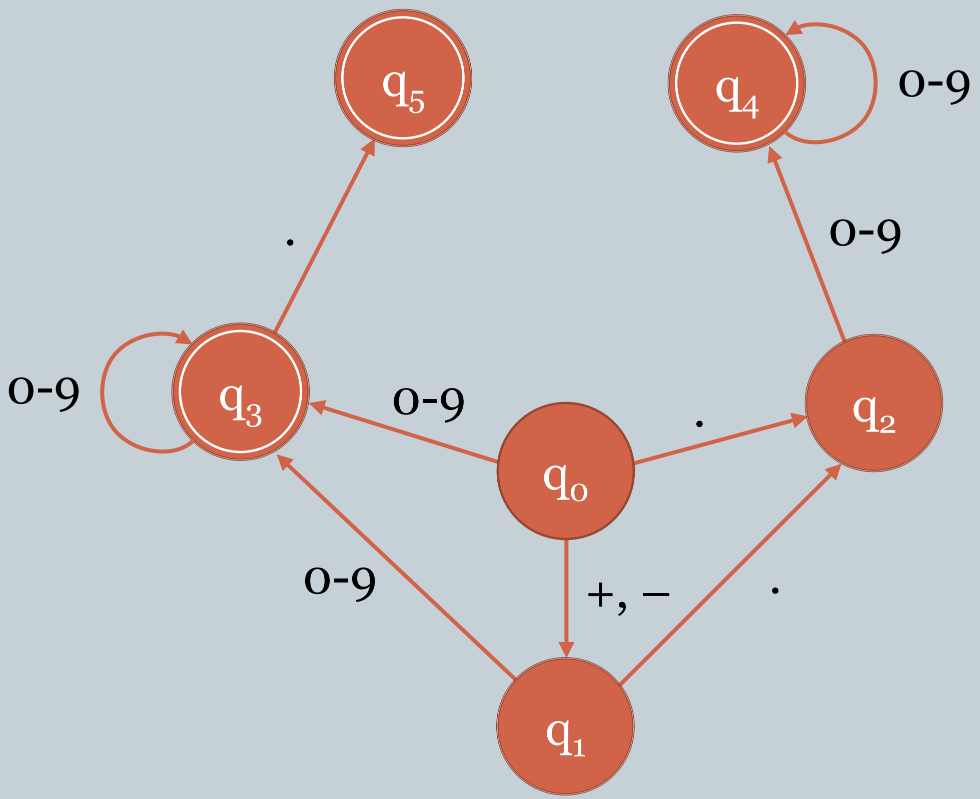
Определение функции переходов



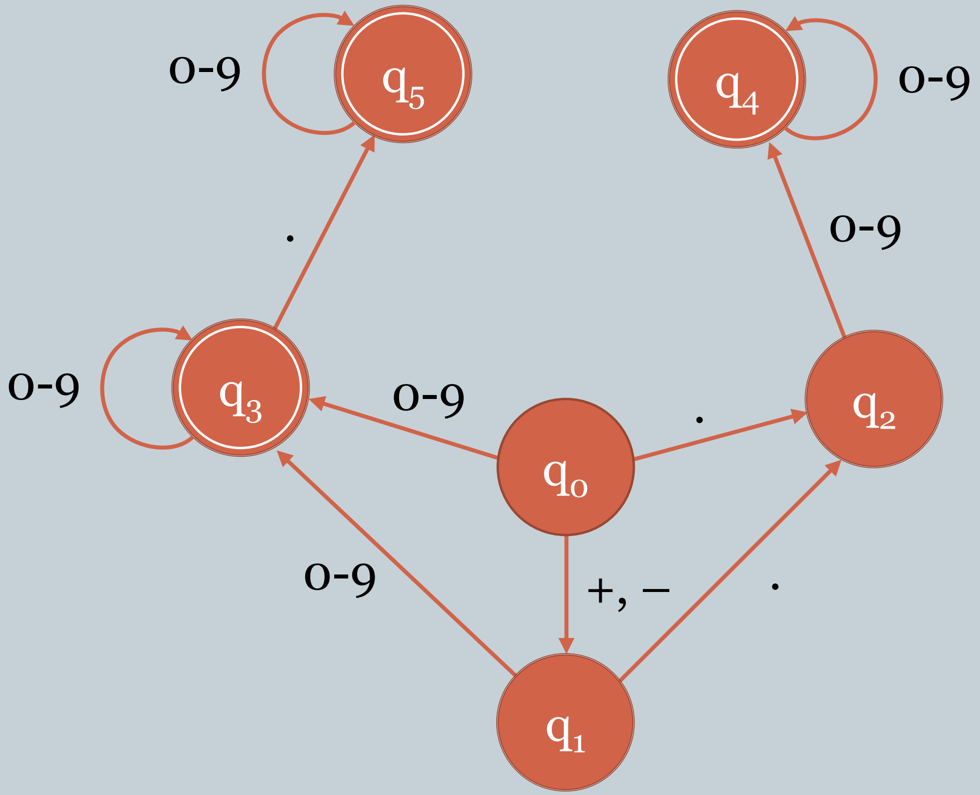
Определение функции переходов



Определение функции переходов



Определение функции переходов



Ввод *диапазонов*:

	+	−	.	0	1	2	3	4	5	6	7	8	9	⊥
q_0														
…														
q_n														

	+, −	.	0-9	⊥
q_0				
…				
q_n				

Ввод *диапазонов*:

	Σ_1	Σ_2	…	Σ_m	$\perp$
q_0					
…					
q_n					

$$\Sigma_i \in \Sigma,\ i = 1, 2, \ldots, m;$$

$$\Sigma_i \cap \Sigma_j = \varnothing,\ i \neq j;$$

$$\Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_m = \Sigma.$$

Определение функции переходов

Табличное представление функции переходов:

	+, −	.	0-9	⊥
q_0	q_1	q_2	q_3	
q_1		q_2	q_3	
q_2			q_4	
q_3		q_5	q_3	HALT
q_4			q_4	HALT
q_5			q_5	HALT

Определение функции переходов

Минимизация ДКА:

	+, –	.	0-9	⊥
q_0	q_1	q_2	q_3	
q_1		q_2	q_3	
q_2			q_4	
q_3		q_4	q_3	HALT
q_4			q_4	HALT

Определение функции переходов

Результат:

	+, −	.	0-9	⊥
q_0	q_1	q_2	q_3	
q_1		q_2	q_3	
q_2			q_4	
q_3		q_4	q_3	HALT
q_4			q_4	HALT

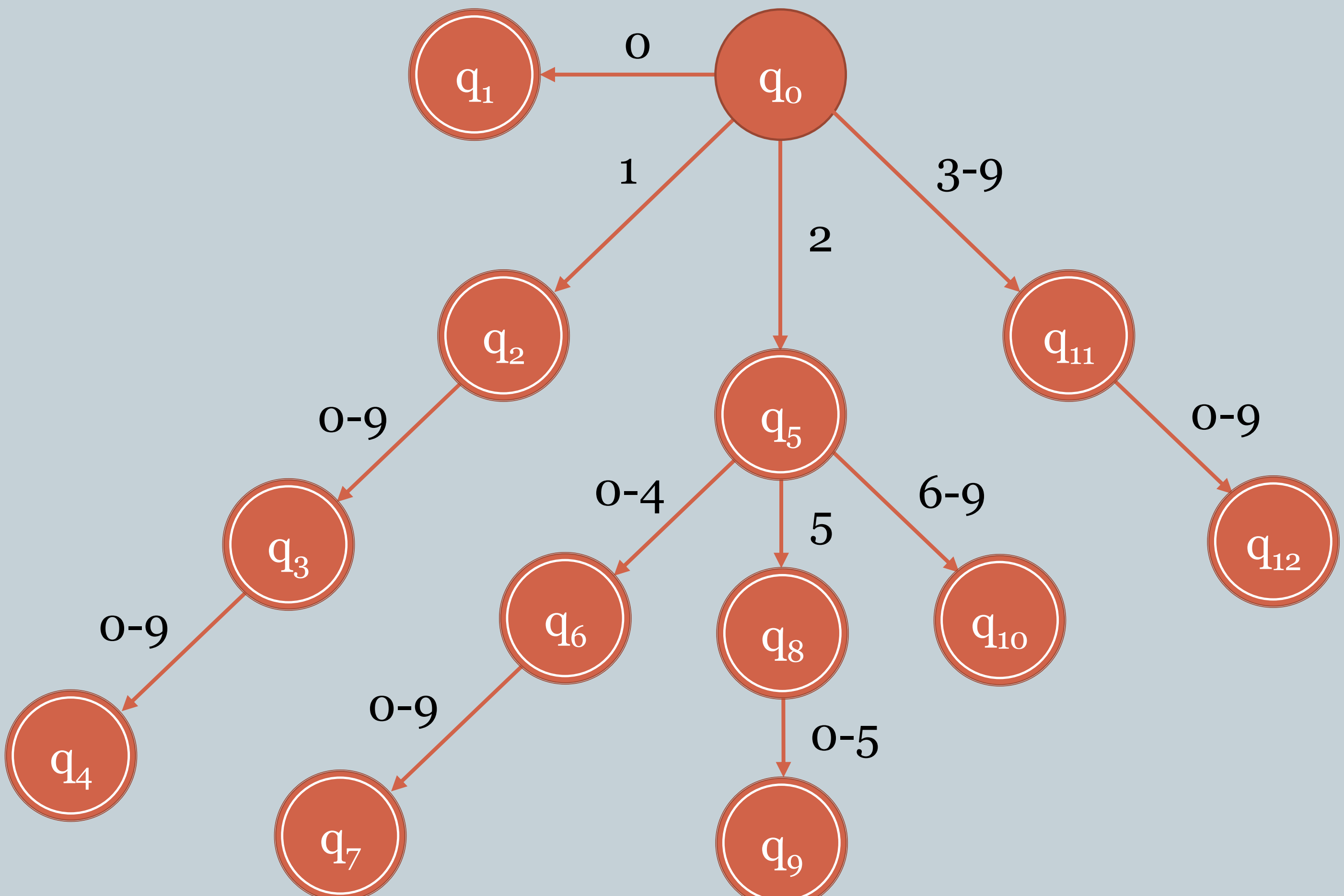
Алфавит языка $\Sigma = \{+, -, ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} = \{+, -, ., 0\text{-}9\}$.

Использование диапазонов

Пример. Язык L описывает десятичные числа в диапазоне от 0 до 255, без ведущих нулей.

Алфавит языка $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} = \{0\text{-}9\}$.

Использование диапазонов



Использование диапазонов

$\Sigma_1 = \{0\}$

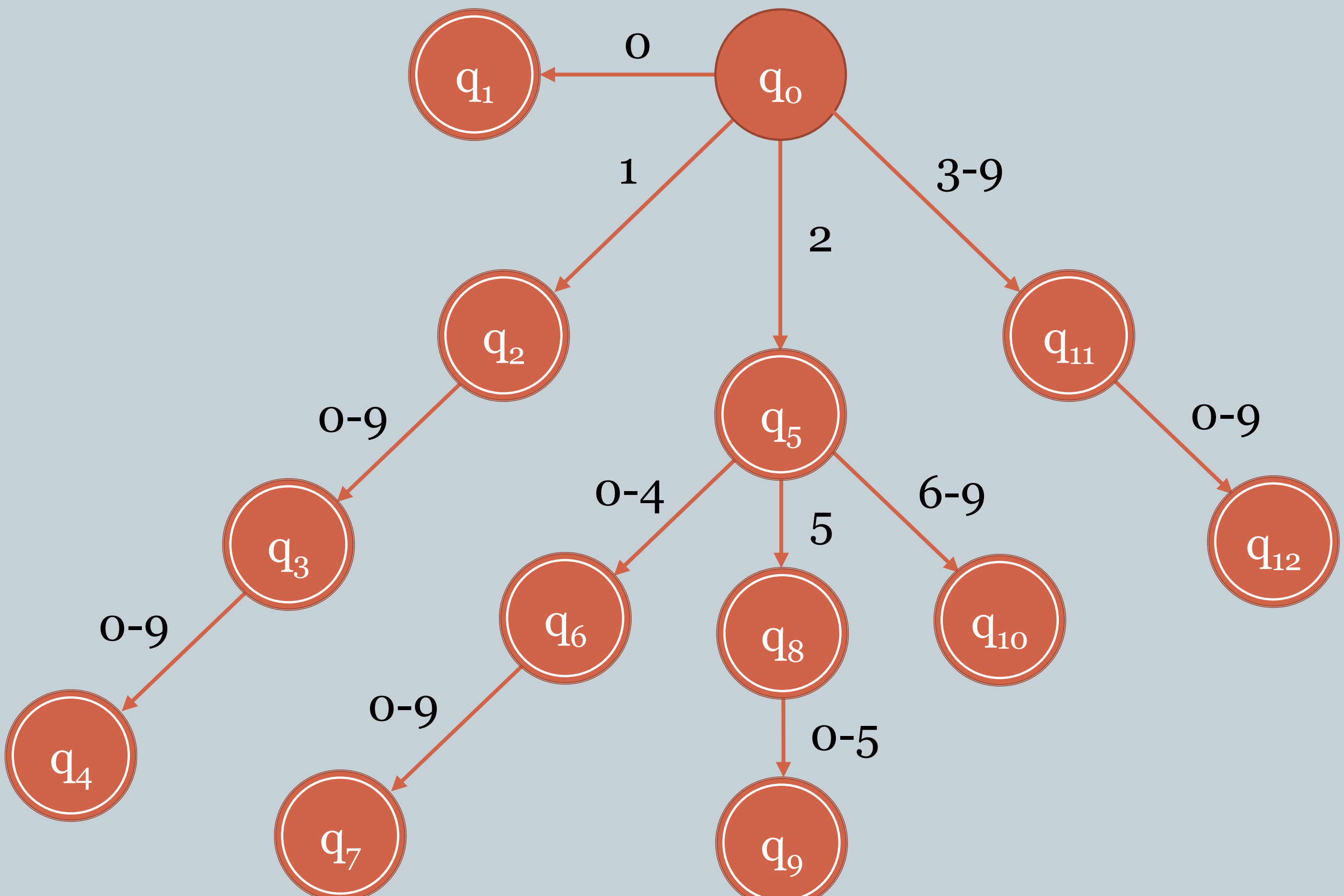
$\Sigma_2 = \{1\}$

$\Sigma_3 = \{2\}$

$\Sigma_4 = \{3\text{-}4\}$

$\Sigma_5 = \{5\}$

$\Sigma_6 = \{6\text{-}9\}$



Использование диапазонов

Пример. Язык L описывает десятичные числа в диапазоне от 0 до 255, без ведущих нулей.

Алфавит языка $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} = \{0\text{-}9\}$.

$$\Sigma_1 = \{0\}, \Sigma_2 = \{1\}, \Sigma_3 = \{2\}, \Sigma_4 = \{3\text{-}4\},$$

$$\Sigma_5 = \{5\}, \Sigma_6 = \{6\text{-}9\}$$

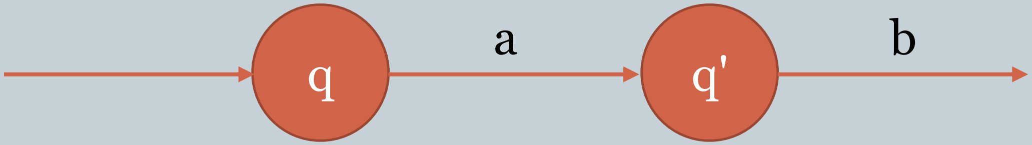
$$\Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4 \cup \Sigma_5 \cup \Sigma_6 = \{0\text{-}9\} = \Sigma.$$

	0	1	2	3-4	5	6-9	$\perp$
q_0							
…							
q_{12}							

Часто используемые приёмы

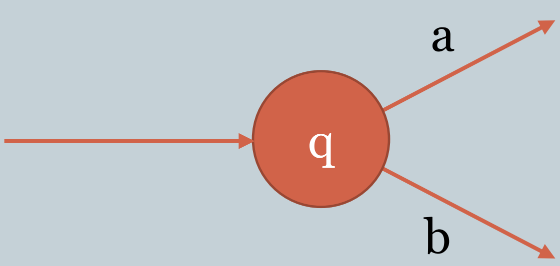
Приём 1. Конкатенация

...ab...



Приём 2. Объединение

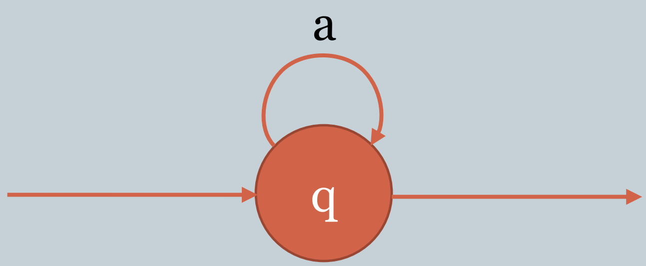
...a... или ...b...



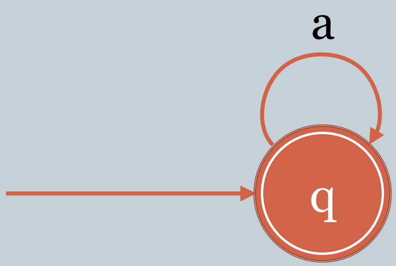
Часто используемые приёмы

Приём 3. Итерация

...e..., ...a..., ...aa..., ...aaa... и т.д.



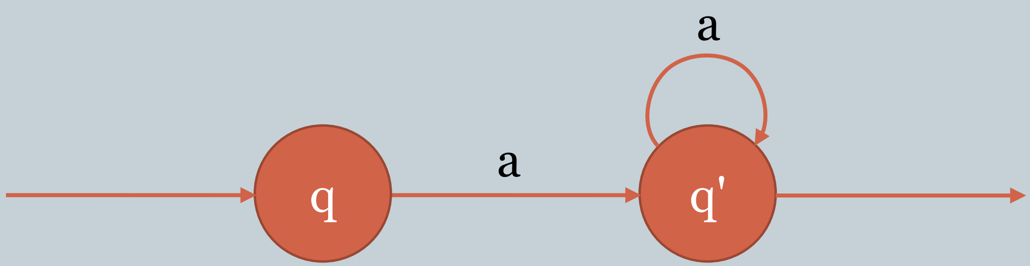
или $q \in F$:



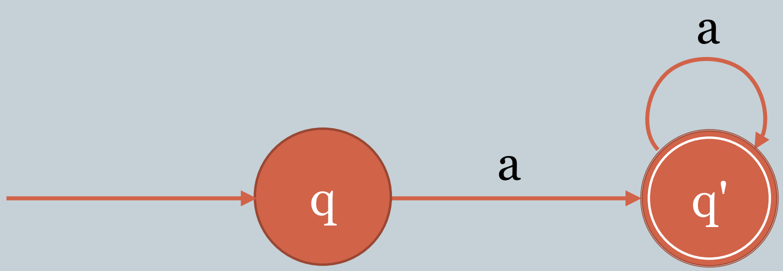
Часто используемые приёмы

Приём 4. Положительная итерация

…a…, …aa…, …aaa… и т.д.



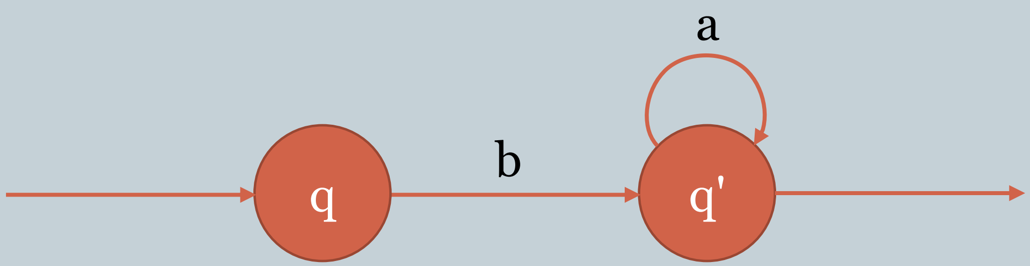
или $q' \in F$:



Часто используемые приёмы

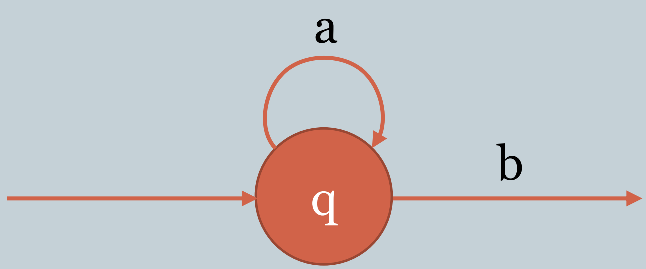
Приём 5. Конкатенация с итерацией

...b..., ...ba..., ...baa... и т.д.



или $q' \in F$;

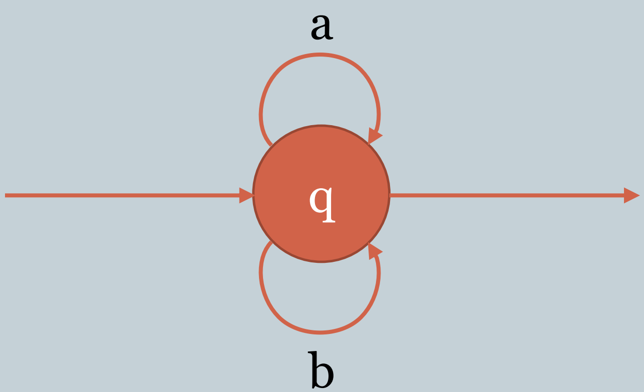
...b..., ...ab..., ...aab... и т.д.



Часто используемые приёмы

Приём 6. Итерация объединения

…*e*…, …a…, …b…, …aa…, …ab…, …ba…, …bb… и т.д.

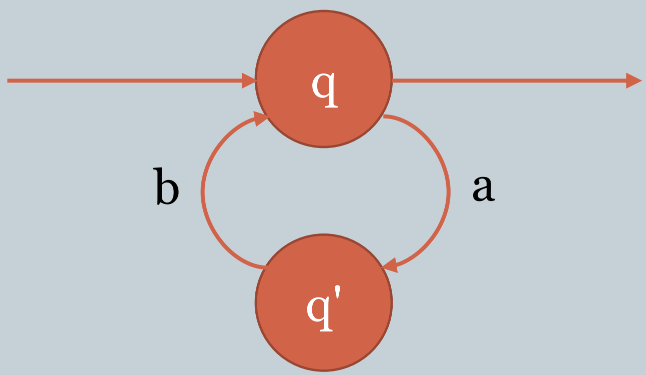


или $q \in F$.

Часто используемые приёмы

Приём 7. Итерация конкатенации

…e…, …ab…, …abab…, …ababab… и т.д.



или $q \in F$.