2. Детерминированные конечные автоматы

Разделы:

- ДКА
- Язык ДКА
- Проверка эквивалентности ДКА
- Минимизация ДКА

Детерминированные КА

$$A = \left(Q, \Sigma, \delta, q_0, F\right)$$

- Q – конечное множество состояний
- Σ – конечное множество входных символов
- $p = \delta(q,a)$ – функция переходов
  - q – состояние
  - a – входной символ (или сигнал)
  - Значение функции – новое состояние
- q_0 – начальное состояние
- F – множество заключительных (допускающих) состояний

Детерминированные КА

- Предположим $a_1a_2...a_n$ – это последовательность входных символов, и ДКА начинает работу в состоянии q_0
- Чтобы найти состояние, в которое ДКА перейдет после обработки a_1, нужно обратиться к функции δ
- Допустим, $\delta(q_0,a_1) = q_1$
- Аналогично определяются состояния $q_2,\dots,q_n$, где $\delta(q_{i-1},a_i) = q_i$ для каждого $_i$
- Если q_n принадлежит множеству F, то входная строка $a_1a_2...a_n$ **допускается**, в противном случае – **отвергается**

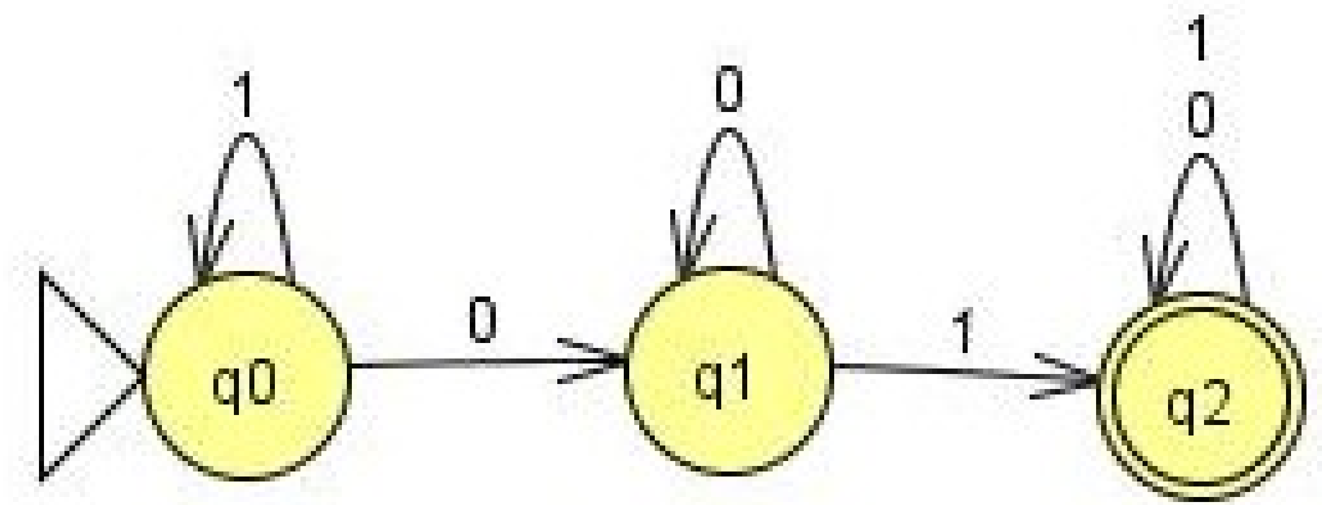
Детерминированные КА

- Для языка
  $L_A = \{x01y \mid x$ и y – строки из 0 и 1$\}$
- Примеры строк этого языка: 01, 11010, 1011101
- Не принадлежат этому языку, например, следующие строки: 0, 1, 1100 и ε
- Алфавит входных символов языка
  $\Sigma = \{0, 1\}$
- q_0 – начальное состояние
- $\delta(q_0,1) = q_0$, $\delta(q_0,0) = q_1$, $\delta(q_1,0) = q_1$, $\delta(q_1,1) = q_2$, $\delta(q_2,0) = \delta(q_2,1) = q_2$

Простые представления КА

- **Диаграмма переходов** - граф, подобный приводившимся ранее
  - любому состоянию из Q соответствует некоторая вершина
  - пусть $\delta(q,a) = p$ для некоторого q из Q и входного символа q из Σ. Тогда диаграмма должна содержать дугу из q в p, отмеченную a (возможно несколько дуг)
  - диаграмма содержит стрелку в начальное состояние
  - вершины, соответствующие заключительным состояниям отмечаются двойным кружком

Простые представления КА



Простые представления КА

- **Таблица переходов**, дающая табличное представление функции δ, из которой очевидны состояния и входной алфавит
- Двум аргументам ставится в соответствие одно значение
- Строки таблицы соответствуют состояниям, столбцы – входным символам
- На пересечении строк для состояния q и столбца для символа a находится состояние $p = \delta(q,a)$

Простые представления КА

	0	1
$\to q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

Простые представления КА

$$\hat{\delta}(p, w)$$

- Это расширенная функция переходов
- Она ставит состоянию *q* и строке *w* (*w*=*xa*; *a* – последний символ в *w*; *x* – все остальное) в соответствие новое состояние *p*, в которое КА попадает из состояния *q*, обработав входную последовательность *w*

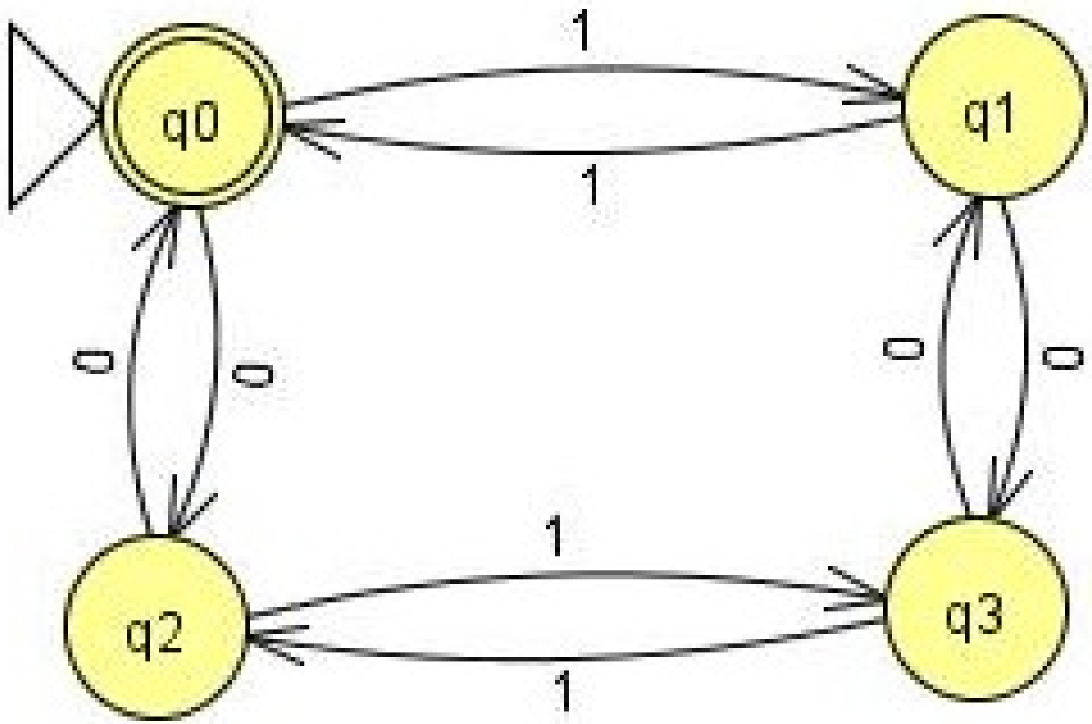
$$\hat{\delta}(q, \varepsilon) = q \qquad \hat{\delta}(q, w) = \delta\left(\hat{\delta}(q, x), a\right)$$

Простые представления КА

- $L = \{w \mid w$ содержит четное число 0 и четное число 1$\}$
- Состояния:
  - q_0 – прочитано четное число 0 и четное число 1.
  - q_1 – прочитано четное число 0 и нечетное число 1.
  - q_2 – прочитано нечетное число 0 и четное число 1.
  - q_3 – прочитано нечетное число 0 и нечетное число 1

$$A = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_0\})$$

Простые представления КА



Простые представления КА

	0	1
*->q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Простые представления КА

- Пусть на вход подается строка 110101
- **СТУДЕНТАМ**: докажите неформально, принадлежит ли она языку?
- Мы ожидаем, что

$$\hat{\delta}(q_0, 110101) = q_0$$

- Результат:

$$\hat{\delta}(q_0, \varepsilon) = q_0$$

$$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \varepsilon), 1) = \delta(q_0, 1) = q_1$$

$$\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$$

$$\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_1, 0) = q_2$$

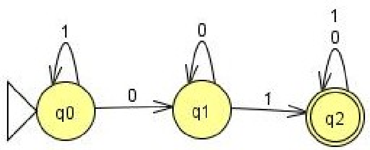
$$\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_3$$

$$\hat{\delta}(q_0, 11010) = \delta(\hat{\delta}(q_0, 1101), 0) = \delta(q_3, 0) = q_1$$

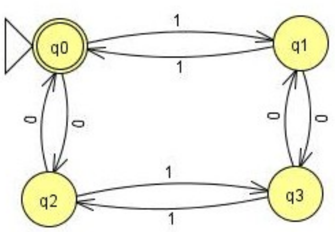
$$\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$$

Язык ДКА

- $L(A)$ = { w | $\hat{\delta}(q_0, w)$ принадлежит F }
- Если язык L есть $L(A)$ для некоторого автомата A, то говорят, что L является **регулярным языком**
- если A – следующий ДКА



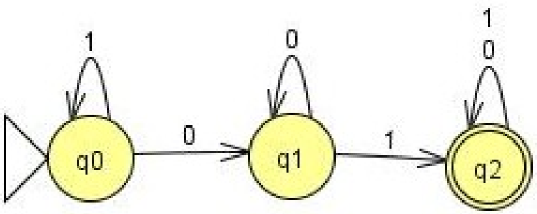
- то $L(A)$ – это множество строк из 0 и 1, которые содержат подстроку 01
- Если A – следующий ДКА



- то $L(A)$ – это множество строк из 0 и 1, которые содержат четное число 0 и четное число 1

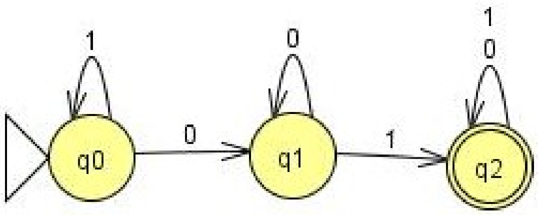
Программная реализация ДКА

```
#include <stdio.h>

#define TOTAL_STATES          3
#define FINAL_STATES          1
#define ALPHABET_CHARCTERS    2

#define UNKNOWN_SYMBOL_ERR    0
#define NOT_REACHED_FINAL_STATE 1
#define REACHED_FINAL_STATE   2

enum DFA_STATES {q0, q1, q2};   // The set Q
enum input {_0, _1};

int  g_Accepted_states[FINAL_STATES] = { q2 };      // The set F
char g_alphabet[ALPHABET_CHARCTERS] = {'0', '1'};   // The set Sigma
int  g_Transition_Table[TOTAL_STATES][ALPHABET_CHARCTERS] = {}; //
Transition function
int  g_Current_state = q0; // Start state of DFA
void SetDFA_Transitions()
{
  g_Transition_Table[q0][_0] = q1;
  g_Transition_Table[q0][_1] = q0;
  g_Transition_Table[q1][_0] = q1;
  g_Transition_Table[q1][_1] = q2;
  g_Transition_Table[q2][_0] = q2;
  g_Transition_Table[q2][_1] = q2;
}
...
```



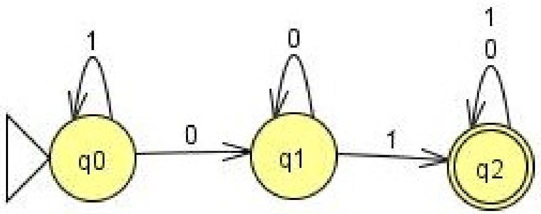
Программная реализация ДКА

```
...
int DFA(const char current_symbol)
{
  int i, pos;
  for (pos = 0; pos < ALPHABET_CHARCTERS; ++pos)
    if (current_symbol == g_alphabet[pos])
      break;     // stops if any character other than 0 or 1
    if (ALPHABET_CHARCTERS == pos)
      return UNKNOWN_SYMBOL_ERR;
    for (i = 0; i < FINAL_STATES; ++i)
    {
      g_Current_state = g_Transition_Table[g_Current_state][pos];
      if (g_Current_state == g_Accepted_states[i])
        return REACHED_FINAL_STATE;
    }
  return NOT_REACHED_FINAL_STATE;
}
...
```



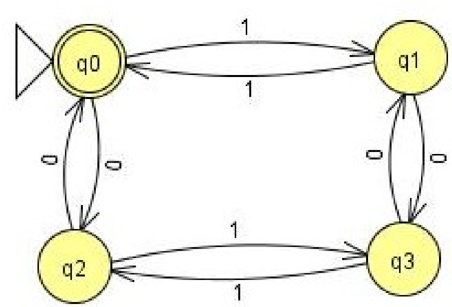
Программная реализация ДКА

```
...
int main(void)
{
  char current_symbol;
  int  result;
  SetDFA_Transitions();    // Fill transition table
  printf("Enter a string with '0' s and '1's:\nPress Enter Key to stop\n");
  while ((current_symbol = getchar()) != '\n' && current_symbol != EOF)
  {
    result = DFA(current_symbol);
    if (REACHED_FINAL_STATE != result && NOT_REACHED_FINAL_STATE != result)
    {
      break;
    }
  }
  if (REACHED_FINAL_STATE == result)
  {
    printf("Accepted");
  }
  else
  {
    printf("Rejected");
  }
  printf("\n\n");
  return 0;
}
```



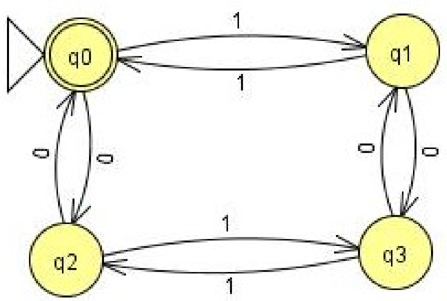
Программная реализация ДКА

```
#include <stdio.h>
#define TOTAL_STATES           4
#define FINAL_STATES           1
#define ALPHABET_CHARCTERS     2
#define UNKNOWN_SYMBOL_ERR     0
#define NOT_REACHED_FINAL_STATE 1
#define REACHED_FINAL_STATE    2

enum DFA_STATES {q0, q1, q2, q3};   // The set Q
enum input {_0, _1};

int  g_Accepted_states[FINAL_STATES] = { q0 };      // The set F
char g_alphabet[ALPHABET_CHARCTERS] = {'0', '1'};  // The set Sigma
int  g_Transition_Table[TOTAL_STATES][ALPHABET_CHARCTERS] = {}; //
Transition function
int  g_Current_state = q0; // Start state of DFA
void SetDFA_Transitions()
{
  g_Transition_Table[q0][_0] = q2;
  g_Transition_Table[q0][_1] = q1;
  g_Transition_Table[q1][_0] = q3;
  g_Transition_Table[q1][_1] = q0;
  g_Transition_Table[q2][_0] = q0;
  g_Transition_Table[q2][_1] = q3;
  g_Transition_Table[q3][_0] = q1;
  g_Transition_Table[q3][_1] = q2;
}
...
```



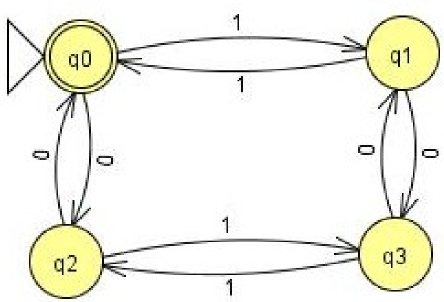
Программная реализация ДКА

```
...
int DFA(const char current_symbol)
{
  int i, pos;
  for (pos = 0; pos < ALPHABET_CHARCTERS; ++pos)
    if (current_symbol == g_alphabet[pos])
      break;     // stops if any character other than 0 or 1
  if (ALPHABET_CHARCTERS == pos)
    return UNKNOWN_SYMBOL_ERR;
  for (i = 0; i < FINAL_STATES; ++i)
  {
    g_Current_state = g_Transition_Table[g_Current_state][pos];
    if (g_Current_state == g_Accepted_states[i])
      return REACHED_FINAL_STATE;
  }
  return NOT_REACHED_FINAL_STATE;
}
...
```



Программная реализация ДКА

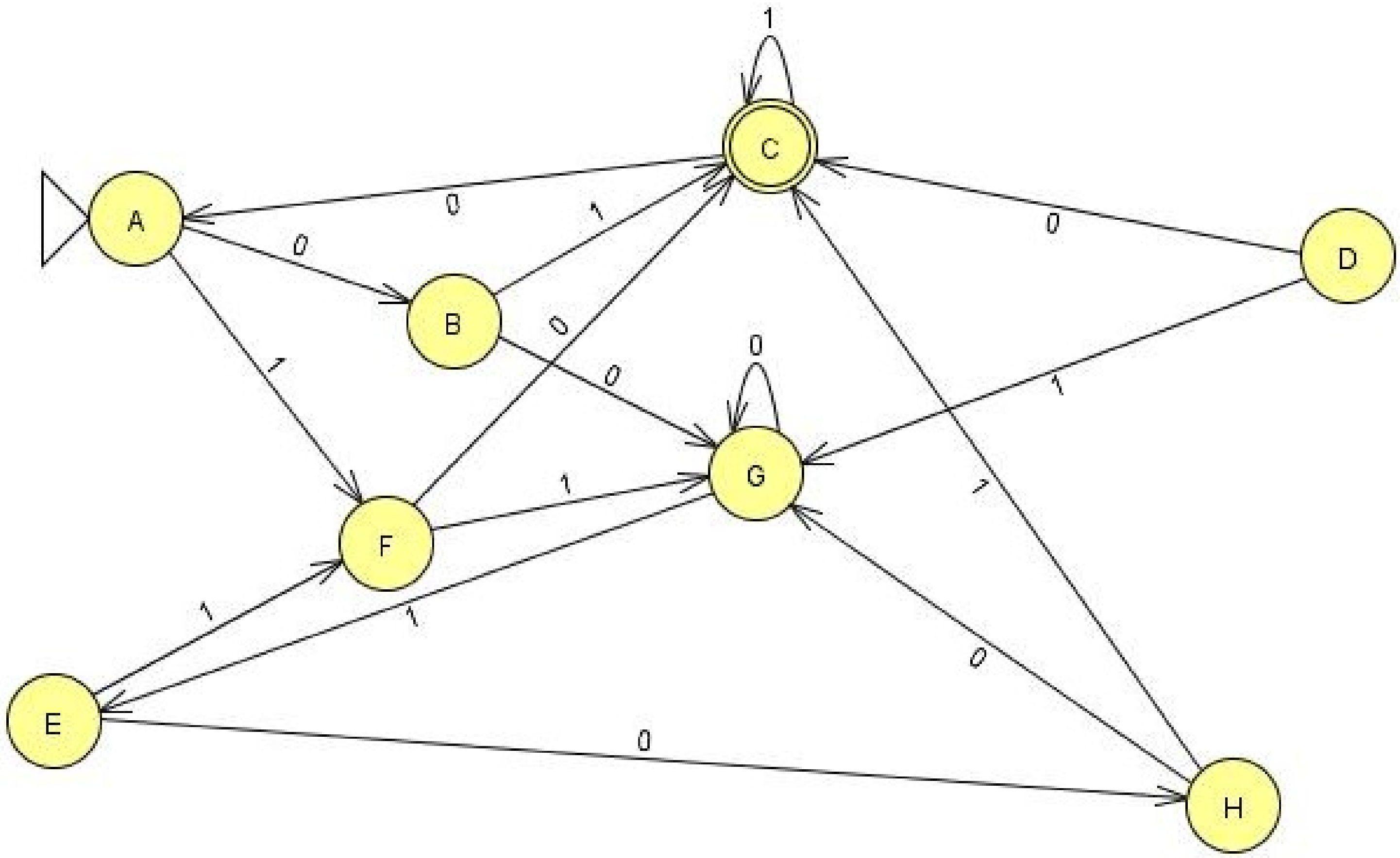
```
...
int main(void)
{
  char current_symbol;
  int  result;
  SetDFA_Transitions();    // Fill transition table
  printf("Enter a string with '0' s and '1's:\nPress Enter Key to stop\n");
  while ((current_symbol = getchar()) != '\n' && current_symbol != EOF)
  {
    result = DFA(current_symbol);
    if (REACHED_FINAL_STATE != result && NOT_REACHED_FINAL_STATE != result)
    {
      break;
    }
  }
  if (REACHED_FINAL_STATE == result)
  {
    printf("Accepted");
  }
  else
  {
    printf("Rejected");
  }
  printf("\n\n");
  return 0;
}
```



Проверка эквивалентности состояний ДКА

- Состояния *p* и *q* **эквивалентны**, если для всех входных строк *w* состояние $\hat{\delta}(p, w)$
- является заключительным тогда и только тогда, когда состояние $\hat{\delta}(q, w)$
- является заключительным
- Неэквивалентные состояния **различимы**, т.е. существует по меньшей мере одна строка, для которой одно из состояний заключительное, а другое – нет

Проверка эквивалентности состояний ДКА

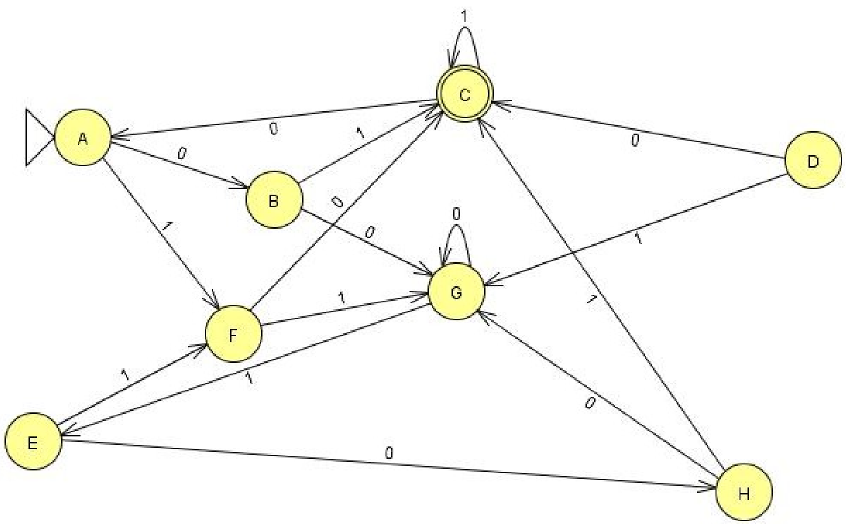


Проверка эквивалентности состояний ДКА

- **Алгоритм заполнения таблицы** производит рекурсивное обнаружение пар различимых состояний ДКА
  1. если состояние *p* – заключительное, а *q* – не заключительное, то пара состояний {*p*, *q*} различима
  2. Пусть *p* и *q* – состояния, по входному символу *a* переходящие в различимые состояния, тогда эта пара различима по простой причине
- **СТУДЕНТАМ**: по какой причине различимы состояния

$$r = \delta(p, a) \qquad s = \delta(q, a)$$

Проверка эквивалентности состояний ДКА



B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G

Проверка эквивалентности состояний ДКА

- **Теорема 2.1:** Если два состояния не различаются с помощью алгоритма заполнения, то они *эквиваленты*
- **Доказательство:** Допустим, утверждение теоремы неверно, и существует, по меньшей мере, одна пара состояний (p, q), для которой выполняются два условия:
  - Состояния p и q различимы
  - Алгоритм заполнения таблицы не может обнаружить различимость p и q (это называется «**плохой парой**»)

Проверка эквивалентности состояний ДКА

- **Доказательство** (продолжение):
- Допустим, у нас есть плохие пары
- Среди них должны быть различимые с помощью кратчайших строк, различающих плохие пары
- Пусть пара $\{p, q\}$ – плохая, $w = a_1a_2...a_n$ – кратчайшая из всех строк, различающих p и q
- Тогда заключительным будет только одно из следующих состояний:

$$\hat{\delta}(p, w) \qquad \hat{\delta}(q, w)$$

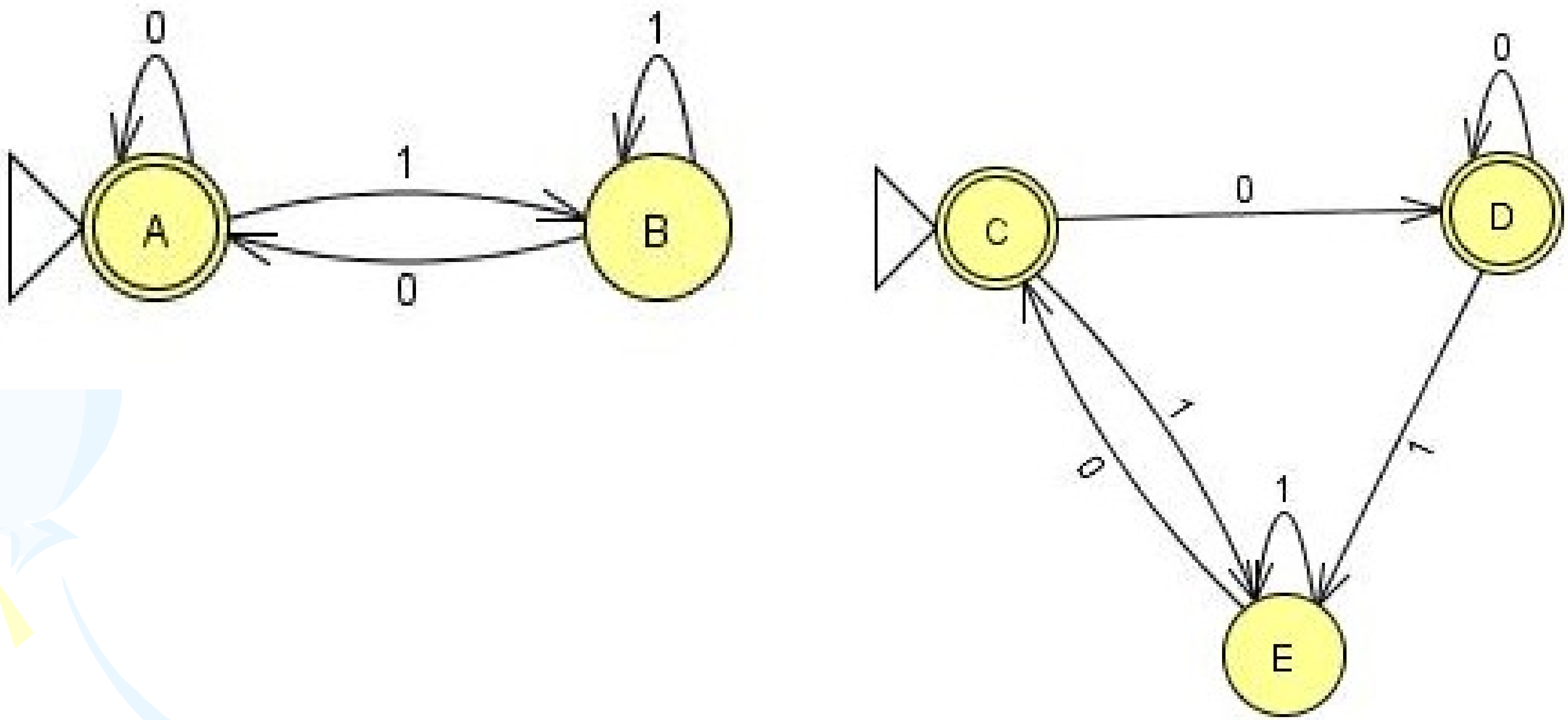
Проверка эквивалентности состояний ДКА

- **Доказательство** (продолжение):
- Рассматриваем следующие состояния
  $r = \delta(p, a_1)$ $s = \delta(q, a_1)$
- Их можно различить строкой $a_2 \ldots a_n$, т.к. она переводит КА из состояний r и s в следующие состояния $\hat{\delta}(p, w)$ $\hat{\delta}(q, w)$
- Строка, отличающая r от s, короче любой строки, различающей плохую пару
- Значит, $\{r, s\}$ – не плохая пара

Проверка эквивалентности состояний ДКА

- **Доказательство** (окончание):
- Однако индуктивная часть алгоритма заполнения не остановится, пока не придет к выводу, что состояния *p* и *q* различимы
- Мы пришли к противоречию с предположением о наличии плохих пар, т.е. их нет
- Следовательно, любую пару различимых состояний можно обнаружить алгоритмом заполнения, и наша **теорема доказана**

Проверка эквивалентности состояний ДКА



СТУДЕНТАМ: докажите формально их эквивалентность

Минимизация ДКА

- Основная идея заключается в том, что эквивалентность состояний позволяет объединять их в блоки
  1. Все состояния в блоке эквивалентные
  2. Любые два состояния из разных блоков неэквивалентны
- В таблице со слайда 18 состояния разбиваются на блоки следующим образом:
  - $\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}$
- Каждая пара эквивалентных состояний помещается в отдельный блок, а состояния, отличные от других, образуют отдельные блоки

Минимизация ДКА

- **Теорема 2.2:** эквивалентность состояний транзитивна, т.е. если для некоторого ДКА состояние *p* эквивалентно *q*, а состояние *q* эквивалентно *r*, то состояния *p* и *r* также эквивалентны
- **Доказательство.** Предположим, что {*p*, *q*} и {*q*, *r*} являются парами эквивалентных состояний, а пара {*p*, *r*} – различима
- Тогда должна существовать строка *w*, для которой является заключительным одно из состояний:

$$\hat{\delta}(p, w) \quad \hat{\delta}(r, w)$$

Минимизация ДКА

- **Доказательство** (продолжение)
- Допустим заключительным является первое из двух названных состояний
- Предполагая заключительным состояние $\hat{\delta}(q, w)$
- Тогда пара {*q*, *r*} – различима (**СТУДЕНТАМ**: почему?)
- Если последнее состояние не заключительное, то пара {*p*, *q*} различима
- Пришли к противоречию, следовательно, *p* и *q* эквивалентны

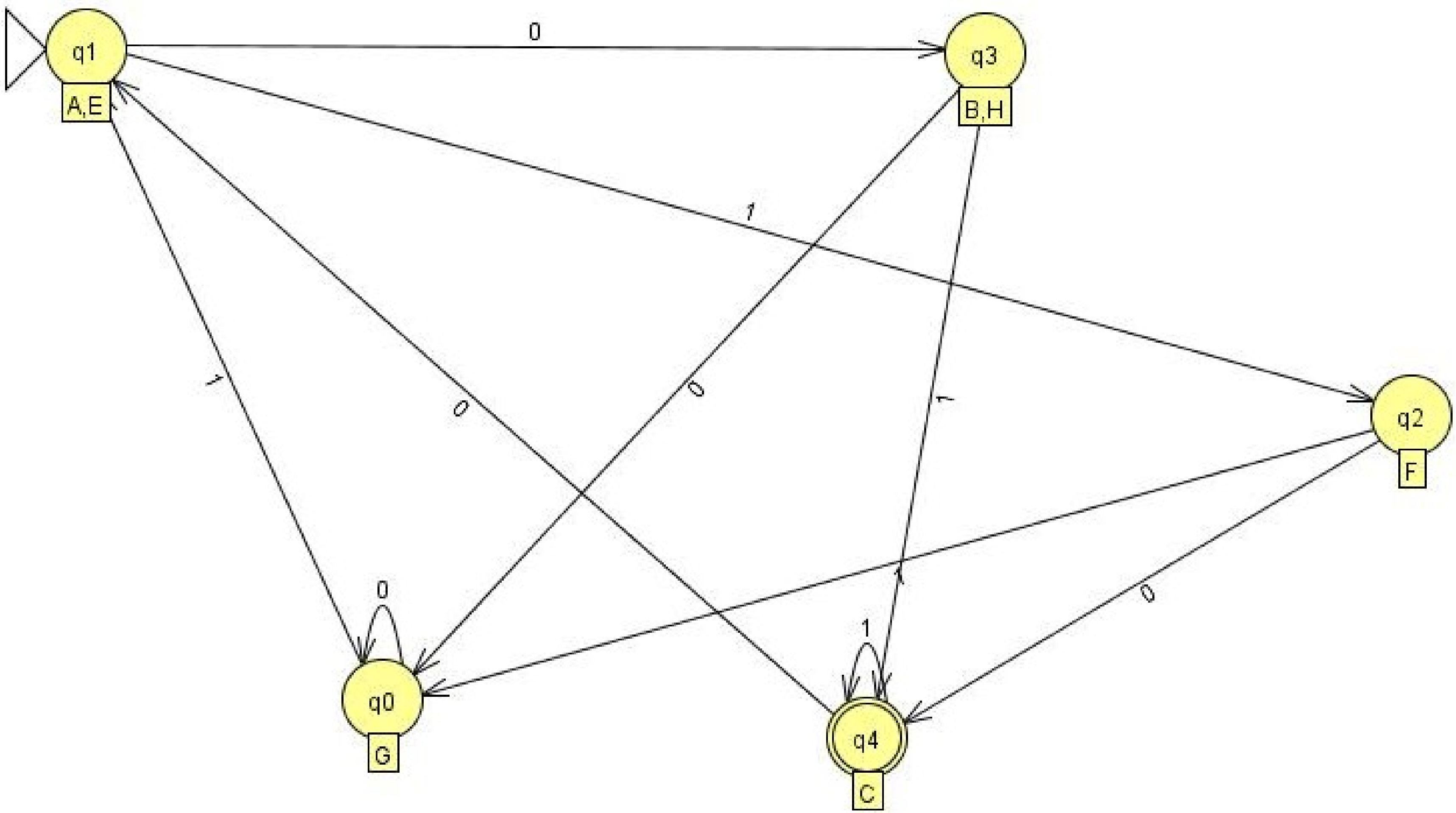
Минимизация ДКА

- **Теорема 2.3:** если для любого состояния ДКА создать блок, который состоит из этого состояния и эквивалентных ему, то различные блоки образуют **разбиение множества состояний**
- Значит, любое состояние может принадлежать только одному блоку
- Состояния в блоке эквивалентны, а в разных блоках – неэквивалентны
- *Без доказательства*

Минимизация ДКА

- Алгоритм минимизации ДКА *A:*
  1. Для выявления всех пар эквивалентных состояний используем алгоритм заполнения таблицы
  2. Для получения блоков взаимно эквивалентных состояний применяем к множеству *Q* алгоритм разбиения
  3. Получаем ДКА *B* с минимальным числом состояний, используя результаты шага (2)
- Пусть γ – функция переходов автомата *B*, *S* – множество эквивалентных состояний ДКА *A*, *a* – некоторый входной символ
- Тогда должен существовать один блок *T*, содержащий $\gamma(q, a)$ для всех состояний *q* из *S*
- Если это не так, то *a* переводит *p* и *q* из *S* в состояния из разных блоков согласно теореме с предыдущего слайда
- Отсюда, состояния *p* и *q* не эквивалентны и не могут принадлежать *S* и можно определять функцию переходов $\gamma(S, a) = T$

Минимизация ДКА



Дополнительные источники

- Гилл, А. Введение в теорию конечных автоматов / А. Гилл. – М.: Наука, 1966. – 272 с.
- Кузнецов, А.С. Теория вычислительных процессов [Текст] : учеб. пособие / А. С. Кузнецов, М. А. Русаков, Р. Ю. Царев ; Сиб. федерал. ун-т. - Красноярск: ИПК СФУ, 2008. – 184 с.
- Короткова, М.А. Математическая теория автоматов. Учебное пособие / М.А. Короткова. – М.: МИФИ, 2008. – 116 с.
- Молчанов, А. Ю. Системное программное обеспечение. 3-е изд. / А.Ю. Молчанов. – СПб.: Питер, 2010. – 400 с.
- Пример реализации конечных автоматов на языке С++ - http://www.devexp.ru/2011/02/konechnye-avtomaty-v-c/

Дополнительные источники

- Теория автоматов / Э. А. Якубайтис, В. О. Васюкевич, А. Ю. Гобземис, Н. Е. Зазнова, А. А. Курмит, А. А. Лоренц, А. Ф. Петренко, В. П. Чапенко // Теория вероятностей. Математическая статистика. Теоретическая кибернетика. — М.: ВИНИТИ, 1976. — Т. 13. — С. 109–188. — URL http://www.mathnet.ru/php/getFT.phtml?jrnid=intv&paperid=28&what=fullt&op
- Серебряков В. А., Галочкин М. П., Гончар Д. Р., Фуругян М. Г. Теория и реализация языков программирования — М.: МЗ-Пресс, 2006 г., 2-е изд. - http://trpl7.ru/t-books/TRYAP_BOOK_Details.htm
- Finite State Machine Generator - http://sourceforge.net/projects/genfsm/
- Введение в схемы, автоматы и алгоритмы - http://www.intuit.ru/studies/courses/1030/205/info