МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ КОНЦЕПЦИИ КОНЕЧНОГО АВТОМАТА

Учебно-методическое пособие для вузов

Составители:
Е.Е. Михайлова,
Г.Э. Вощинская,
Е.М. Михайлов

Воронеж
Издательский дом ВГУ
2016

Оглавление

Формализованный (формальный) язык — язык, характеризующийся точными правилами построения выражений и их понимания. Он строится в соответствии с четкими правилами, обеспечивая непротиворечивое, точное и компактное отображение свойств и отношений изучаемой предметной области (моделируемых объектов).

В отличие от естественных языков формальным языкам присущи четко сформулированные правила семантической интерпретации и синтаксического преобразования используемых знаков, а также то, что смысл и значение знаков не изменяется в зависимости от каких-либо прагматических обстоятельств (например, от контекста) [1].

КОНЕЧНЫЙ АВТОМАТ

Конечный автомат — это инструмент (алгоритм) для распознавания строк (цепочек символов) какого-либо (формального) языка [2, 3].

Конечные автоматы широко используются на практике, например, в синтаксических и лексических анализаторах, тестировании программного обеспечения на основе моделей и т.д. Конечный автомат идеально подходит для реализации искусственного интеллекта в играх, получая аккуратное решение без написания громоздкого и сложного кода.

Формально конечный автомат определяется пятью характеристиками:

$$\mathrm{M} = (K, \sum, \delta, S, F),$$

где

1) $\sum$ — *входной алфавит* (конечное множество *входных символов*), из которого формируются *входные слова*, воспринимаемые конечным автоматом;
2) K — конечное *множество состояний*;
3) δ — множество (функция Вики) *переходов*;
4) S — начальное состояние ($S \in K$);
5) F — множество *заключительных состояний* ($F \subseteq K$).

Принято полагать, что конечный автомат начинает работу в начальном состоянии, последовательно считывая по одному символу строки. Считанный символ переводит автомат в новое состояние, зависящее от текущего символа и функции переходов.

Читая входную строку и делая переходы из состояния в состояние, автомат после прочтения последнего символа строки окажется в некотором состоянии.

Если это состояние является заключительным, то о строке говорят, что она принадлежит языку, принимаемому автоматом (автомат *принимает* строку). В противном случае строка не принадлежит языку, принимаемому автоматом.

Функцию переходов можно представить также в виде диаграммы состояний и таблицы переходов.

- Диаграмма состояний (или граф переходов) — графическое представление множества состояний и функции переходов. Представляет собой размеченный ориентированный граф, вершины которого — состояния, дуги — переходы из одного состояния в другое, а метки дуг — символы, по которым осуществляется переход из одного состояния в другое. Если переход из одного состояния в другое может быть осуществлен по одному из нескольких символов (недетерминированный автомат), то все они должны быть надписаны над дугой графа.
- Таблица переходов — табличное представление функции δ. Обычно в такой таблице каждой строке соответствует одно состояние, а столбцу — один допустимый входной символ. В ячейке на пересечении строки и столбца записывается состояние, в которое должен перейти автомат, если в данном состоянии он считал данный входной символ.

ŋqdbyeY bⱻ debⱻbYзe d eaαbcoy Зqdd Yqyè

Детерминированным конечным автоматом называется такой автомат, в котором нет дуг с меткой ε (строка, не содержащая ни одного символа), и из любого состояния возможен переход в другое состояние только по одному символу.

Недетерминированный конечный автомат является обобщением детерминированного и здесь рассматриваться не будет.

Задача 1. Распознавание строк

Рассмотрим простую задачу.

Пусть дан конечный автомат $M1 = (\{A, B\},\{0,1\}, \delta, A, A)$, который характеризуется:

1) двумя входными символами: 0 и 1;
2) двумя состояниями: A и B;
3) функцией переходов:
   - $\delta(A, 0) = A$,
   - $\delta(A, 1) = B$,
   - $\delta(B, 0) = B$,
   - $\delta(B, 1) = A$;

   Первые два перехода означают, что при чтении символа «0» в состоянии A автомат остается в том же состоянии, при чтении символа «1» в состоянии A осуществляется переход в состояние B. Два других перехода — аналогично.
4) начальным состоянием A;
5) заключительным состоянием A.

Этот конечный автомат разбирает строки, составленные из символов «0» и «1».

Диаграмма состояний данного конечного автомата представлена на рис. 1.

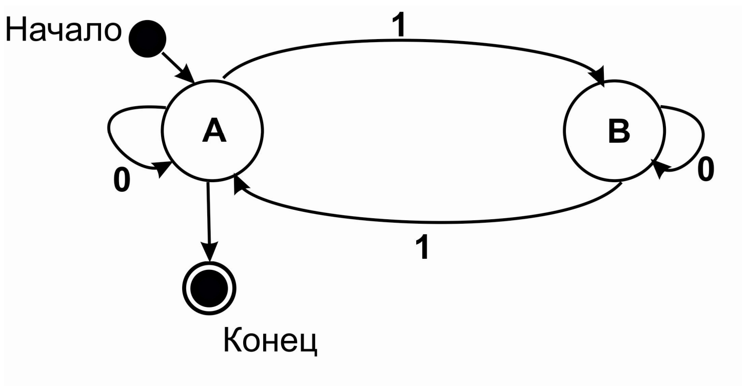

Рис. 1. Диаграмма состояний конечного автомата задачи 1.

Таблица переходов конечного автомата задачи 1 представлена в табл. 1.

Таблица 1

Таблица переходов конечного автомата для задачи 1

	0	1
A	***A***	***B***
B	***B***	***A***

При чтении строки 01001011 автомат последовательно проходит состояния: *A*(начальное), *A*, *B*, *B*, *B*, *A*, *A*, *B*, *A*. Так как заключительным состоянием является *A*, то данная строка *принимается* автоматом.

При чтении строки 00111 автомат проходит состояния: *A* (начальное), *A*, *A*, *B*, *A*, *B*. Поскольку *B* не является заключительным состоянием, строка 00111 не принимается, то есть она не принадлежит языку, принимаемому этим автоматом.

Следующая программа иллюстрирует работу данного конечного автомата.

```
enum state = (stA, stB); // состояния
public bool parse(string s)
  {
    bool res = false;
    state st;
```

```
    if (s != "")  // строка не пустая
    {
        st = state.stA; // начальное состояние А
        int len = s.Length;
        for (int i = 0; i < len; i++)
        {
          switch (st)
          {
            case state.stA:
            {
              if (s[i] == '1') st = state.stB; // пришла 1 – переходим в B
              break;
            }
            case state.stB:
            {
              if (s[i] == '1') st = state.stA; // пришла 1 – переходим в A
              break;
            }
          } // switch
        } // for
        res = st == state.stA; // конечное состояние должно быть А
    }
    return res;
  }
```

Примеры работы программы задачи 1 показаны на рис. 2.

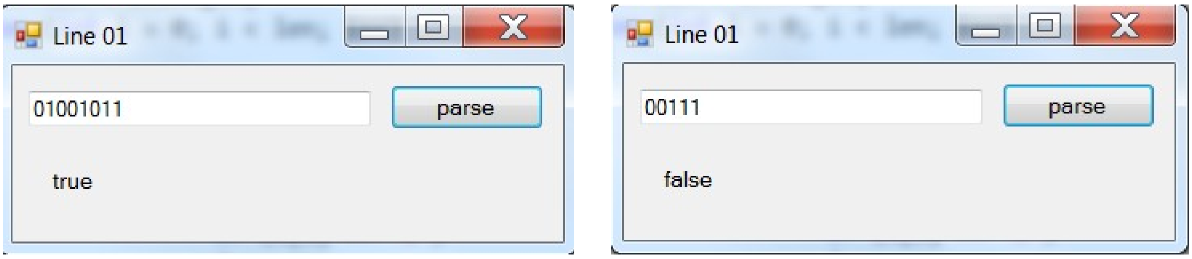

Рис. 2. Примеры работы программы распознавания строк

Лексический анализатор

Первой фазой процесса компиляции является *лексический анализ*. Лексический анализ — процесс аналитического разбора входной последовательности символов (например, такой как исходный код на одном из языков программирования) с целью получения на выходе последовательности символов языка, называемых *токенами* (подобно группировке букв в словах). Группа символов входной последовательности, идентифицируемая на выходе процесса как токен, называется *лексемой*. В процессе лексиче-

ђqdqyэY bЕ dэdЕbYзэ d єcaqcy ЭqddYqyэ