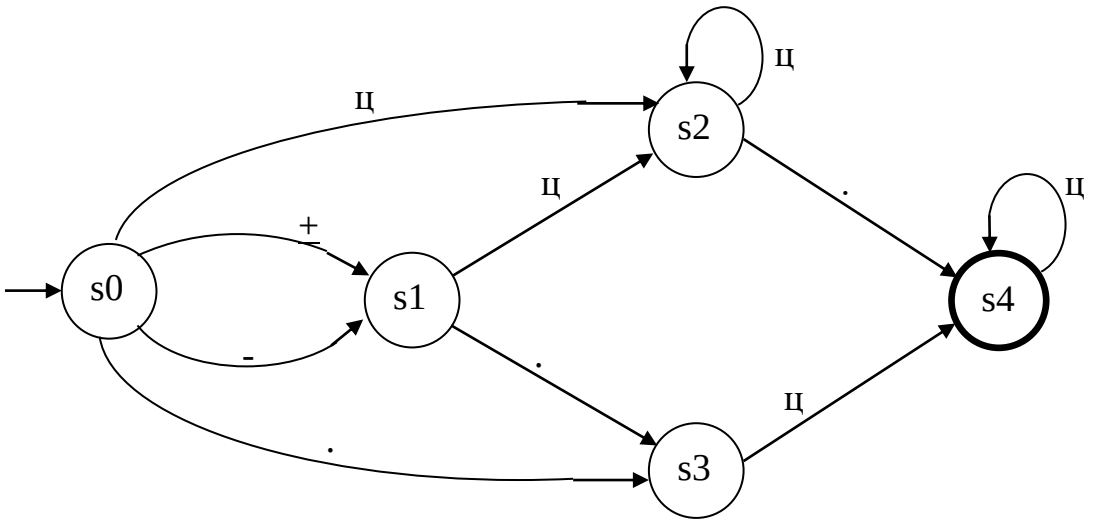
Программная реализация конечных детерминированных распознавателей

Рассмотрим два способа программной реализации конечных детерминированных распознавателей: *компиляционный* (программный) и *интерпретационный* (табличный).

1. Компиляционный способ.

Пусть задан граф конечного детерминированного распознавателя. Его можно преобразовать в программу распознавания цепочки языка, размер которой пропорционален количеству вершин (состояний) и дуг в графе. Каждому состоянию соответствует часть кода: считывание очередного символа и, если считан не концевой маркер, определение состояния перехода. Процесс распознавания заканчивается, если считан концевой маркер или если состоянием перехода является состояние ошибки. При переходе в состояние ошибки может быть выдано сообщение о типе ошибки. Результат распознавания определяется состоянием, в котором завершён процесс распознавания: если оно допускающее -–цепочка допускается, иначе – отвергается.

Рассмотрим два алгоритма реализации следующего конечного детерминированного распознавателя:



Алгоритм 1.

Код каждого состояния отметим соответствующей меткой. Переменную *x* будем использовать для хранения символа входной цепочки. Анализ переменной *x* выполним с помощью оператора множественного выбора. Переход в новое состояние закодируем оператором безусловного перехода на метку. Концевой маркер обозначим символом ‘┤’

Псевдокод алгоритма 1 представлен ниже:

```
s0: ввод (x);
    выбор (x)
        x∈{‘+’,’-‘} : переход на s1;
        x∈{‘0’..’9‘} : переход на s2;
        x = '.' : переход на s3;
        x = ‘┤’ : вывод ( ‘Отвергнуть, цепочка пуста’), конец.
    конец выбор;

s1: ввод (x);
    выбор (x)
        x∈{‘0’..’9‘} : переход на s2;
        x = '.' : переход на s3;
        x∈{‘+’,’-‘} : вывод ( ‘Отвергнуть, недопустимы два знака
                                подряд’), конец.
        x = ‘┤’ : вывод ( ‘Отвергнуть, нет значения константы’), конец.
    конец выбор;

s2: ввод (x);
    выбор (x)
        x∈{‘0’..’9‘} : переход на s2;
        x = '.' : переход на s4;
        иначе : вывод ( ‘Отвергнуть, в вещественной константе
                                должна быть точка ’), конец.
    конец выбор;

s3: ввод (x);
    выбор (x)
        x∈{‘0’..’9‘} : переход на s4;
        иначе : вывод ( ‘Отвергнуть, после точки должна быть цифра’),
                конец.
    конец выбор;

s4: ввод (x);
    выбор (x)
        x∈{‘0’..’9‘} : переход на s4;
        x∈{‘+’,’-‘,'.'} : вывод ( ‘Отвергнуть, последний символ
                                не может встречаться дважды’), конец.
        x = ‘┤’ : вывод ( ‘Допустить’), конец.
    конец выбор;
```

Алгоритм 2.

Номер текущего состояния будем хранить в переменной S, а для его определения будем использовать множественный выбор, каждая альтернатива которого соответствует состоянию. Символ входной цепочки будем хранить в переменной x, а для его анализа будем использовать множественный выбор. Переходу в новое состояние соответствует присваивание переменной S нового значения. Распознавание будем выполнять в цикле, пока не будет введён концевой маркер или пока в S не записано состояние ошибки. Состояние ошибки представим отрицательным числом, которое так же является кодом ошибки. После выхода из цикла выводится результат – ‘допустить’ или ‘отвергнуть’ и, если ‘отвергнуть’, то выводится сообщение об ошибке.

Псевдокод алгоритма 2 представлен ниже:

```
S := 0;
повторять
    ввод (x);
    выбор (S)
      S = 0 :  выбор (x)
                  x∈{‘+’,’-‘} : S := 1;
                  x∈{‘0’..’9‘} : S := 2;
                  x = '.' : S := 3;
                  x = ‘┤’ : S := -1;
               конец выбор;

      S = 1 : выбор (x)
                  x∈{‘0’..’9‘} : S = 2;
                  x = '.' : S = 3;
                  x∈{‘+’,’-‘} :  S := -2;
                   x = ‘┤’ : S = -3;
               конец выбор;

      S = 2 : выбор (x)
                  x∈{‘0’..’9‘} : S = 2;
                  x = '.' : S = 4;
                  иначе : S = -4;
               конец выбор;

      S = 3 : выбор (x)
                  x∈{‘0’..’9‘} : S = 4;
                  x = '.' : S = 4;
                  иначе : S = -5;
               конец выбор;
```

```
            S = 4 : выбор (x)
                        x∈{‘0’..’9‘} : S = 4;
                        x∈{‘+’,’-‘,'.'} : S = -6;
                    конец выбор;
        конец выбор;
пока  S ≥ 0  и  x ≠ ‘┤’ ;

выбор (S)
    S = 4 : вывод ( ‘Допустить’);
    S = -1 : вывод ( ‘Отвергнуть, цепочка пуста’);
    S = -2 : вывод ( ‘Отвергнуть, недопустимы два знака подряд’);
    S = -3 : вывод ( ‘Отвергнуть, нет значения константы’);
    S = -4 : вывод ( ‘Отвергнуть, в вещественной константе
                        должна быть точка ’);
    S = -5 : вывод ( ‘Отвергнуть, после точки должна быть цифра’);
    S = -6 : вывод ( ‘Отвергнуть, последний символ не может
                        встречаться дважды’);
конец выбор;
```

2. Интерпретационный способ.

В этом способе реализации используется табличный способ задания конечного детерминированного распознавателя. Таблица распознавателя сохраняется в памяти и обрабатывается по определённому, общему для всех распознавателей, алгоритму. Номер состояния распознавателя будем хранить в переменной S, символ входной цепочки - в переменной x. На каждом шаге распознавания, пока не введён концевой маркер, определяется состояние перехода путём обращения к элементу таблицы распознавателя, расположенному в строке, соответствующей входному символу x, и столбце, соответствующем состоянию S. С помощью функции $L(x)$ поставим в соответствие каждому входному символу номер строки таблицы. Для быстрого вычисления $L(x)$ представим эту функцию в табличной форме и сохраним в массиве L таким образом, что элемент $L[x]$ содержит номер строки таблицы распознавателя, соответствующей входному символу x. В результате, для вычисления значения функции $L(x)$ достаточно обратиться к элементу $L[x]$. Переходы в состояние ошибки закодируем отрицательными числами, представляющими собой код ошибки. Сообщения об ошибках будем хранить в таблице ошибок. Выход из процесса распознавания происходит при вводе концевого маркера или достижении состояния ошибки.

Итак, для реализации конечного детерминированного распознавателя интерпретационным способом, необходимо иметь:

1) таблицу переходов конечного детерминированного распознавателя;
2) множество допускающих состояний;
3) массив L для определения строки, таблицы, соответствующей входному символу;
4) таблицу ошибок.

Ниже представлен псевдокод алгоритма интерпретационного способа реализации конечного детерминированного распознавателя.

```
S := 0;
повторять
      ввод (x);
      S := таблица_переходов[L[x],S];
пока  S ≥ 0  и  x ≠ '┤ ';

если S ∈ множество_допускающих_сотояний
     то  вывод ('Допустить')
иначе  вывод('Отвергнуть ', таблица_ошибок[|S|]);
```

Этот алгоритм является универсальным и позволяет получать программы реализации различных распознавателей путём изменения обрабатываемых данных.