УДК 681.3

РАЗРАБОТКА ФОРМЫ ПРЕДСТАВЛЕНИЯ МОДИФИЦИРОВАННЫХ ПРОДУКЦИОННЫХ АЛГОРИТМОВ В ЗАДАЧАХ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ

© 2012 В. М. Довгаль[1], О. Ф. Корольков[2] А. А Чаплыгин[3], В. О. Королькова[4]

[1]докт. техн. наук, профессор каф. программного обеспечения и администрирования информационных систем
e-mail: vmdovgal@yandex.ru

Курский государственный университет

[2]канд. техн. наук, доцент каф. программного обеспечения вычислительной техники,
e-mail: oleg_korolkov@mail.ru
[3]канд. техн. наук, доцент каф. программного обеспечения вычислительной техники,
e-mail: alex_chaplygin@mail.ru
[4]инженер каф. программного обеспечения вычислительной техники
e-mail: viktoria_korolkova@mail.ru

Юго-Западный государственный университет

Рассматривается форма представления модифицированной универсальной продукционной системы для реализации способа и программы автоматической генерации кода программы по заданному продукционному алгоритму в системах управления.

Ключевые слова: автомат, состояние, алгоритм, продукция, программа, код, управление, генерация кода.

Введение

На протяжении двадцати лет ряд авторов [Туккель, Шалыто 2002] развивают направление, называемое «автоматное программирование», или «программирование от состояний», или «программирование с явным выделением состояний», с целью разработки инструментальных средств автоматической генерации кодов программ, основанной на расширенной модели конечных автоматов и ориентированной на создание широкого класса приложений в интересах реализации процессов управления сложными объектами. При этом на основе анализа предметной области выделяются источники входных воздействий для системы управления и механизмы, которые при заданных условиях формируют управляющие воздействия на объект. Механизмы управления реализуются одним автоматом или множеством взаимодействующих автоматов.

Методика преобразования автомата Мура в продукционный алгоритм

На основе алгоритмической системы А. А. Маркова одним из авторов данной статьи была разработана продукционная универсальная алгоритмическая система быстрых преобразований символьных данных[Довгаль 1996; Довгаль 2009]. Продукционные алгоритмы представляют собой конечный список продукций и переходов на их множестве, а каждая из продукций в общем случае задается словом вида

$$M1:{}'\gamma i \rightarrow \beta j\ {}' :M2, \qquad (1)$$

где γi (образец), βj (модификатор) – цепочки символов заданного рабочего алфавита, $M1$, $M2$ – метки (натуральные числа) перехода на определенные номера продукций из их списка в алгоритме, *!* (маркер) – указатель позиции в обрабатываемом слове, с которой необходимо выполнять сопоставление при обнаружении продукцией очередной позиции вхождения образца в пространстве обрабатываемого слова, а сопоставление должно быть продолжено.

Построим продукционный алгоритм, моделирующий работу автомата Мура (см. [Королькова и др. 2012]), представленного на рисунке 1.

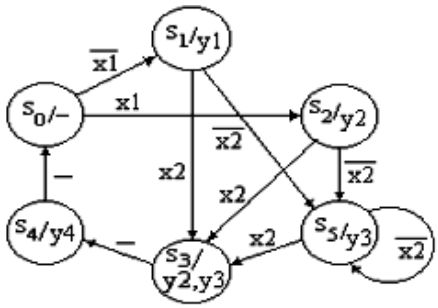

Рис. 1. Граф автомата Мура

Произведем замену букв входного и выходного алфавита следующими:

входные: $x1$ – $Z1$, $\bar{x}1$ – $Z2$, $x2$ – $Z3$, $\bar{x}2$ – $Z4$, «–» – $Z5$ (входной сигнал отсутствует);

выходные: $y1$ – $Y1$, $y2$ – $Y2$, $y3$ – $Y3$, ($y2$, $y3$) – $Y4$, $y4$ – $Y5$, «–» – $Y6$ (выходной сигнал отсутствует);

состояния: S_0, S_1, S_2, S_3, S_4, S_5.

В указанной работе предложен нормальный алгоритм А.А. Маркова, реализующий заданный автомат Мура, который состоит из трех блоков и приводится ниже.

Первый блок композиции:

1 $\rightarrow^* S_0$

Второй блок композиции:

2 $S_0Z1 \rightarrow Y1S_1$
3 $S_0Z2 \rightarrow Y2S_2$
4 $S_1Z4 \rightarrow Y3S_5$ (2)
5 $S_1Z3 \rightarrow Y4S_3$
6 $S_2Z3 \rightarrow Y3S_3$
7 $S_2Z4 \rightarrow Y3S_5$
8 $S_3Z4 \rightarrow Y4S_3$
9 $S_3Z5 \rightarrow Y4S_4$
10 $S_4Z5 \rightarrow Y6S_0$
11 $S_5Z3 \rightarrow Y3S_3$
12 $S_5Z4 \rightarrow Y3S_5$

Третий блок композиции

13 $\varphi \rightarrow^*$.

Служебный символ «*» в продукции с номером 13 продукционного алгоритма (2) обозначает продукцию, завершающую свою работу после однократного срабатывания, а φ – алфавитная переменная, принимающая свои значения на алфавите внутренних состояний автомата.

На вход алгоритма поступают слова, представляющие собой последовательности букв, отражающие состояния объекта управления, а на выходе алгоритм формирует управляющие слова, переводящие объект управления в целевое состояние.

Приведенный нормальный алгоритм, моделирующий работу управляющего автомата Мура, работает следующим образом. Пусть один из вариантов входного обрабатываемого слова будет иметь вид

$$Z2Z4Z4Z3Z5Z5\ldots\ldots\ldots \quad (3)$$

Тогда вначале однократно выполнит свою работу первая продукция алгоритма, являющаяся единственной в продукционном алгоритме (2), составляющем первый блок композиции, и будет осуществлен переход на второй блок композиции этого алгоритма, а слово будет иметь следующий вид:

$$\underline{S_0 Z2} Z4Z4Z4Z3Z5Z5. \quad (4)$$

Теперь будут сравниваться левые части продукций алгоритма (2) и в случае совпадения подчеркнутой части слова (4) будет проведена замена на правую часть продукции, после чего обрабатываемое слово будет иметь вид

$$Y2\underline{S_2\ Z4}Z4Z4Z3Z5Z5 \quad (5)$$

Далее работа продолжится и будет осуществляться сопоставление с подчеркнутой частью слова (5), выделяющей позицию вхождения образца. Необходимо отметить, что каждый раз после очередного срабатывания для дальнейшего сопоставления будет выполняться активация продукций блока два, до тех пор пока не будет найдена соответствующая продукция.

Нами предлагается модифицированный продукционный алгоритм [Довгаль 1996], в котором в блоке два после срабатывания очередной продукции происходит переход по метке. Первая метка в начале продукции является указателем перехода на следующую продукцию в случае несовпадения фрагмента слова и левой части (образца) предшествующей продукции при сопоставлении, а метка в конце продукции – это указатель, по которому происходит переход на номер следующей продукции. Тогда модифицированный продукционный алгоритм будет иметь следующий вид:

2	$3 : S_0Z1 \rightarrow Y1S_1 : 4$
3	$14 : S_0Z2 \rightarrow Y2S_2 : 6$
4	$5 : S_1Z4 \rightarrow Y3S_5 : 10$
5	$14 : S_1Z3 \rightarrow Y4S_3 : 8$
6	$7 : S_2Z3 \rightarrow Y3S_3 : 8$
7	$14 : S_2Z4 \rightarrow Y3S_5 : 11$
8	$9 : S_3Z4 \rightarrow Y4S_3 : 8$
9	$14 : S_3Z5 \rightarrow Y4S_4 : 10$
10	$14 : S_4Z5 \rightarrow Y6S_0 : 2$
11	$12 : S_5Z3 \rightarrow Y3S_3 : 8$
12	$14 : S_5Z4 \rightarrow Y3S_5 : 12$

Предлагаемый модифицированный продукционный алгоритм работает быстрее (по числу затрачиваемых шагов), чем канонический марковский алгоритм (2), в котором каждый раз при очередном сопоставлении с обрабатываемым словом происходит перебор продукций до тех пор, пока не произойдет очередное совпадение ее образца и соответствующего фрагмента обрабатываемого слова.

Используя язык программирования C#, запишем текст программы, которая непосредственно преобразует форму представления продукционного алгоритма в исходный код программы, моделирующей работу управляющего автомата без меток:

```
private void button1_Click(object sender, EventArgs e)
    {
      String s = "s0z2z4z4z4z3z5z5";

      String[] val = {"s0z1", "s0z2", "s2z3",
"s1z4","s1z3","s2z4","s3z4","s3z5","s4z5","s5z3","s5z4"};
      String[] new_val = {"y1s1",
"y2s2","y3s3","y3s5","y4s3","y3s5","y4s3","y4s4","y6s0","y3s3","y3s5"};
      int i;
      while (true) {
        for (i=0; i < val.Length; i++) {
          if (s.Contains(val[i])) {
            s = s.Replace(val[i], new_val[i]);
            break;
          }
        }
        if (i == val.Length)
        {
          textBox1.Text = s;
          return;
        }
      }
```

Программа выполнена с использованием двух функций: *Contains()*, которая выдает *true*, если обнаружено вхождение текущего образца в строке (3), и *Replace()*, которая производит замену обнаруженного образца на модификатор, расположенный в правой части каждой продукции алгоритма (2).

Построим более быструю программу на языке C, позволяющую моделировать работу автомата для продукционного алгоритма с метками перехода между продукциями во время работы такого алгоритма.

Продукционный алгоритм задается в виде текстового файла:

```
1: s0z0 -> y1s1 :3
6: s0z2 -> y2s1:3
6: s1z1 -> y1s2: 4
6: s2z2 -> y2s2 :4,
```

где символ ‘:’ – разделитель тела продукции и метки перехода на соответствующий номер продукции. Если число в метке больше числа продукций, то произойдет выход из продукционного алгоритма по ошибке.

Программа автоматической генерации кода программы по заданному продукционному алгоритму:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXSTR 1024

typedef struct {
  int m1;
  char from[10];
  char to[10];
  int m2;
} production;

char *replace(char *s, char *w, char *f, char *t)
```

```
{
  char *n = malloc(strlen(s) - (strlen(t) - strlen(f)));
  memcpy(n, s, w - s);
  memcpy(n + (w - s), t, strlen(t));
  memcpy(n + (w - s) + strlen(t), w + strlen(f), strlen(s) - (w - s) -
strlen(f));
  *(n + strlen(s)) = 0;
  return n;
}

int main(int argc, char* argv[])
{
  production *prod = malloc(MAXSTR * sizeof(production));
  production *p = prod;
  int num = 0;
  char *s = malloc(MAXSTR);
  char *ss;
  int i;
  FILE *f;

  if (argc < 2) {
    fprintf(stderr, "Usage prod.exe <file name>\n");
    return 1;
  }
  f = fopen(argv[1], "r");
  while (!feof(f)) {
    fscanf(f, "%i: %s -> %s :%i\n", &p->m1, p->from, p->to, &p->m2);
    ++p;
    ++num;
  }
  scanf("%s", s);
  while (1) {
    i = 0;
    p = prod;
    while (i < num) {
      if (ss = strstr(s, p->from)) {
        s = replace(s, ss, p->from, p->to);
        i = p->m2;
        p = &prod[i];
      } else {
        i = p->m1;
        p = &prod[i];
      }
    }
    if (i >= num)
      break;
  }
  printf("%s\n", s);
  return 0;
}
```

Таким образом, выполнены все необходимые построения для перехода от заданного управляющего автомата Мура к программе, моделирующей работу любого продукционного алгоритма, используемого для целей управления дискретными объектами.

Заключение

В работе предложена реализация технологической схемы «управляющий автомат – алгоритм – код программы», отличающейся от традиционной схемы «алгоритм – “ручной” метод реализации кода программы – исполнение кода». Таким

образом, удаляется сложный и приводящий к ошибкам этап кодирования, связанный с человеческим фактором. Процесс автоматической генерации кода программы уменьшает число ошибок в программе.

Библиографический список

Довгаль В. М. Методы модификации формальных систем обработки символьной информации. Курск, 1996. 113 с.

Довгаль В.М. и др. Инструментальные средства организации быстрых символьных вычислений: краткая история перспективы // Информационно-измерительные и управляющие системы. 2009. № 4. Т. 7. С. 28–32.

Королькова В. О., Корольков О. Ф., Чаплыгин А. А. Довгаль В. М. К вопросу решения проблемы автоматической генерации кода программ по заданному управляющему продукционному алгоритму// В мире научных открытий. 2012. № 1 (25). С. 220–236.

Туккель Н. И., Шалыто А. А. От тьюрингова программирования к автоматному // Мир ПК. 2002. №2. С. 144–149.