КОДОГЕНЕРАЦИЯ ПОСРЕДСТВОМ ГРАФОВ-ПЕРЕХОДОВ

И.А. Тутов, старший преподаватель.,
В.С. Гительман, студент гр. 8Т8Б
Томский политехнический университет
E-mail: mailto:vsg16@tpu.ru

Введение

Программирование программируемых логических контроллеров (ПЛК) осуществляется в основном на языках LD, FBD, ST стандарта ГОСТ Р МЭК 61131-3-2016. Для написания программы на этих языках требуется профильное образование. Вместе с тем существует SFC, который не является самостоятельным средством программирования ПЛК и служит для организации кода. Сложность написания программ приводит к высокой вероятности допущения ошибки или неверного понимания сформулированной задачи между технологом и программистом. Сама методология программирования не побуждает структурировать решаемую проблему и алгоритм. Первоначально решать данную проблему призвана граф-схема алгоритма (ГСА), однако эта методология разработана на этапе становления цифровой вычислительной техники и не удовлетворяет современным требованиям представления алгоритмов [1, с. 80-81].

Решение

Для решения задачи упрощения написания программного кода разработана программа интерпретации детерминированных конечных автоматов (КА) в графической форме в среде программирования Delphi. Задача графического представления конечных автоматов решается с помощью ООП. Конечный автомат (КА) – это модель устройства, которое имеет входы и выходы, при этом в каждый момент времени устройство находится в одном состоянии из множества возможных. При работе КА на вход к нему поступают входные воздействия, а на выходе формируются выходные сигналы.

В программе реализуется создание направленного графа: автомат Мура. Это позволяет упростить задачи взаимодействия заказчика и программиста за счёт разделения их сфер. Программисту не нужно знать, что происходит на производстве того или иного оборудования, а заказчику не нужно разбираться в программе. Простота создания графов и отслеживания переходов из состояния в состояние позволяет добиться понимания заказчиком процессов, происходящих в программе на уровне разработки и утверждения технического задания. Более того, генерация кода упрощает работу программиста, избавляет его от рутинного написания программного кода для того или иного ПЛК, значительно снижает требования к квалификации программиста [2]. Ещё одно явное преимущество КА перед применением языка FBD для проектирования блоковых диаграмм заключается в менее громоздкой структуре схем и удобством внесения корректировок в полученный граф автомата.

В качестве интерфейса при работе с программой используются такие объекты, как состояние и переход. При создании состояния рисуется прямоугольник со скруглёнными углами, атрибутами которого является название данного состояния и действие в нём. При создании второго состояния появляется возможность создавать переход между состояниями, присваивать условие этому переходу. Переход представляет собой стрелку. На рисунке 1 показана модель КА, в которой изображены два состояния и переход между ними.

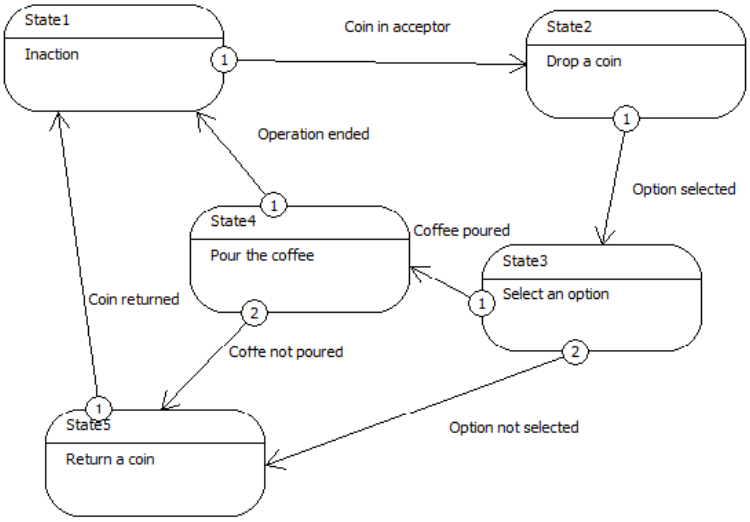

Рис. 1. Граф состояний и переходов

При графическом отображении нескольких выходящих переходов из одного состояния можно задавать приоритетность. Данный параметр учитывает то, какой по счёту будет происходить проверка условия перехода из данного состояния. Реализуется порядковым номером в кружочке в начале стрелки.

В программе имеется возможность при двойном клике на любом объекте, будь то переход, состояние или приоритет, изменять названия, описания, приоритетность и веса, то есть доступно редактирование.

Сочетание клавиш ctrl + a при английской раскладке клавиатуры позволяет создавать переходы после нажатия данных клавиш и щелка левой кнопкой мыши на форме. В то же время аналогичным образом сочетание ctrl + r даёт возможность создавать состояния, при этом не кликая на пункты меню с кнопками «Состояние» и «Переход», что позволяет повысить скорость реализации КА.

Существует также возможность перетаскивания действий или условий, изображённых на переходах. После создания графа можно отобразить его в табличном виде. На рисунке 3 изображена таблица, полученная на основе графа из рисунка 2.

Начальное	Условие	Конечное
State1	Coin in acceptor	State2
State2	Option selected	State3
State3	Coffee poured	State4
State3	Option not selected	State5
State4	Operation ended	State1
State4	Coffe not poured	State5
State5	Coin returned	State1

Рис. 2. Таблица на основе полученного графа

Более того, существует возможность на основе полученного графа сгенерировать код на языке ST и C. Пример сгенерированного кода для рассмотренного автомата приведён на рисунке 3.

```
   N RETURNED : BOOL;
   ION NOT SELECTED : BOOL;
COFFE NOT POURED : BOOL;
END_VAR

CASE STATE OF
STATE1:
  IF COIN IN ACCEPTOR
  THEN
    STATE := STATE2;
  END_IF;


STATE2:
  IF OPTION SELECTED
  THEN
    STATE := STATE3;
  END_IF;


STATE3:
  IF COFFEE POURED
  THEN
    STATE := STATE4;
  END_IF;

  IF OPTION NOT SELECTED
  THEN
    STATE := STATE5;
  END_IF;


STATE4:
  IF OPERATION ENDED
  THEN
    STATE := STATE1;
  END_IF;

  IF COFFE NOT POURED
  THEN
    STATE := STATE5;
  END_IF;


STATE5:
  IF COIN RETURNED
  THEN
    STATE := STATE1;
  END_IF;
END_CASE
```

```
#include <iostream>
using namespace std;

int main() {
  setlocale(LC_ALL, "Russian");
  uint16_t state;
  while(1) {
    switch (state) {
      case STATE1:
      {
        cout <<"Inaction\n";
        if (Coin in acceptor) state = STATE2;
        break;
      }

      case STATE2:
      {
        cout <<"Drop a coin\n";
        if (Option selected) state = STATE3;
        break;
      }

      case STATE3:
      {
        cout <<"Select an option\n";
        if (Coffee poured ) state = STATE4;
        if (Option not selected) state = STATE5;
        break;
      }

      case STATE4:
      {
        cout <<"Pour the coffee\n";
        if (Operation ended) state = STATE1;
        if (Coffe not poured) state = STATE5;
        break;
      }

      case STATE5:
      {
        cout <<"Return a coin\n";
        if (Coin returned) state = STATE1;
        break;
      }

      default:
      {
         cout << "Выполнен переход в неизвестное состояние или в модели не было задано начальное.\n";
         return 0;
      }
    }
  }
  return 0;
}
```

Рис. 3. Сгенерированный код на языке ST и C

Заключение

В результате работы программы можно сделать вывод, что программа построения графов состояний и переходов в полной мере позволяет отображать в наглядном виде данные графы. Существует возможность промежуточного отображения данных в табличном виде. Более того, можно сгенерировать программу на языках C [3] и ST и добавить поддержку любого языка, в том числе ассемблера.

Список использованных источников

1. В.Д. Пароннджанов. Как улучшить работу ума: Алгоритмы без программистов — это очень просто! — М.: Дело, 2001. — 360с.
2. А. А. Шалыто, Использование граф-схем и графов переходов при программной реализации алгоритмов логического управления. II, Автомат. и телемех., 1996, выпуск 7, 144 – 169.
3. Конечный автомат (он же машина состояний) на чистом C // Хабр URL: https://habr.com/ru/post/241941/ (дата обращения: 09.03.2021).