ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ КАК ОДНА ИЗ ПАРАДИГМ ПРОГРАММИРОВАНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Чистяков М. Ю.

Миськевич В. И. – канд. филос. наук, доцент

В статье рассматривается нетрадиционный подход к программированию, основанный на логике предикатов первого порядка. В качестве примера использования данного подхода демонстрируется решение известной загадки Эйнштейна “О пяти домах” на языке логического программирования Prolog.

На сегодняшний день программирование в сознании многих неразрывно связано с понятием **алгоритма**. Алгоритм компьютерной программы – это набор инструкций, описывающих порядок действий ЭВМ для достижения некоторого результата. Такой подход к программированию, когда на программиста ложится задача составления алгоритма решения задачи и занесения его в память ЭВМ, называется **императивным**. Колоссальное распространение императивных языков программирования обусловлено, в первую очередь тем, что они естественным образом соответствуют архитектуре современных ЭВМ. В качестве своей алгоритмической модели современные ЭВМ используют так называемую **машину фон Неймана**, основанную, в свою очередь, на машине Тьюринга, которые базируются на понятии алгоритма.

Если взглянуть на развитие языков программирования, то можно заметить, что с течением времени они становились все ближе к естественному языку. Внесение набора команд в первые ЭВМ представляло собой скорее аппаратный процесс по установке переключателей и перемычек. Затем, после того как была воплощена в жизнь идея фон Неймана о представлении управляющей программы как набора данных в памяти, в качестве языка стали выступать машинные коды. Вскоре появились языки ассемблера, а после – языки высокого уровня, такие как язык **Си**. Однако, хоть языки высокого уровня значительно приблизилось к естественным (яркий пример – условный оператор if, который буквально означает “если”), задача программирования по-прежнему сводилась к построению алгоритма.

Другой подход к решению различного рода задач на ЭВМ предлагает парадигма **декларативного программирования**. Суть данного подхода заключается в том, что алгоритм решения задачи генерируется машиной автоматически, и для решения необходимо формальное описание поставленной задачи. Подвидом декларативной парадигмы принято считать **логическое программирование**. Логическое программирование для формального описания задачи использует **логику предикатов**. Логика предикатов широко используется в математике для записи различных теорем и утверждений и удобна для формализации. Кроме того, законы логического вывода позволяют оперировать формально записанными данными для получения новой информации. Пользуясь этими законами, машина пытается самостоятельно вывести необходимую информацию из данных. Таким образом, задача программиста состоит уже не в разработке алгоритма решения задачи, отвечая на вопрос ‘**как?**’, а в правильной формальной записи входных данных, концентрируясь на том, ‘**что’** ему необходимо получить. Логическое программирование активно применяется в исследованиях искусственного интеллекта.

Одним из первых языков логического программирования является **Prolog**. Название Prolog есть сокращение, означающее “программирование в терминах логики”. Идея использовать логику в качестве основы языка программирования возникла в середине 70-х годов. В 1980-х годах, например, Prolog активно использовался во время разработок японской национальной программы “Компьютеры пятого поколения”.

Программа на языке Prolog состоит из фактов, правил логического вывода и запросов, по которым осуществляется поиск решения задачи, пользуясь механизмом **поиска с возвратом** и **унификацией**. Получив запрос, программа пытается найти описание этого запроса в своём коде. Затем, найдя факт, она производит унификацию и возвращает ответ по запросу. Встретив правило, машина преобразует исходный запрос в набор других запросов, необходимых для удовлетворения первого и пытается решить их.

Рассмотрим применение данного языка на конкретном примере. В качестве задачи возьмём известную загадку Эйнштейна. Вот её формулировка:

1) На улице стоят пять домов
2) Англичанин живёт в красном доме
3) У испанца есть собака
4) В зелёном доме пьют кофе
5) Украинец пьёт чай
6) Зелёный дом стоит сразу справа от белого дома
7) Тот, кто курит Old Gold, разводит улиток
8) В жёлтом доме курят Kool
9) В центральном доме пьют молоко
10) Норвежец живёт в первом доме
11) Сосед того, кто курит Chesterfield, держит лису
12) В доме по соседству с тем, в котором держат лошадь, курят Kool
13) Тот, кто курит Lucky Strike, пьёт апельсиновый сок

14) Японец курит Parliament
15) Норвежец живёт рядом с синим домом

Кто держит зебру?

Автор данной статьи попытался решить задачу самостоятельно в уме. На это ушло примерно 54 минуты☺. Решение данной задачи популярным императивным подходом на одном из популярных языков программирования возможно, но очень затруднительно. В то же время, решение этой задачи с использованием декларативного подхода сводится к грамотному формальному описанию данных и составлению запроса. Ниже приведён пример решения задачи. Даже не вдаваясь в конструктивы языка видно, что программа почти целиком состоит из перечисления входных условий и занимает приблизительно столько же места, сколько её формулировка. Алгоритм поиска ответа формируется машиной самостоятельно, пользуясь механизмом поиска с возвратом и унификацией.

```
%Порядок следования элементов в списке одного дома:
%Цвет, национальность, напиток, сигары, животное
solve(X):-
  HouseList = [_,_,_,_,_],
  add([red,english,_,_,_], HouseList),
  add([_,spanish,_,_,dog], HouseList),
  add([green,_,coffee,_,_], HouseList),
  add([_,ukrainian,tea,_,_], HouseList),
  isRight([white,_,_,_,_], [green,_,_,_,_], HouseList),
  add([_,_,_,old_gold,snail], HouseList),
  add([yellow,_,_,kool,_], HouseList),
  HouseList = [_,_,[_,_,milk,_,_],_,_],
  HouseList = [[_,norwegian,_,_,_],_,_,_,_],
  near([_,_,_,chesterfield,_], [_,_,_,_,fox], HouseList),
  near([_,_,_,_,horse], [_,_,_,kool,_], HouseList),
  add([_,_,juice,lucky_strike,_], HouseList),
  add([_,japanise,_,parliament,_], HouseList),
  near([_,norwegian,_,_,_], [blue,_,_,_,_], HouseList),
  add([_,X,_,_,zebra], HouseList).
%Предикат отношения "справа" для X и Y
isRight(X, Y, [X,Y|_]).
isRight(X, Y, [_|L]):-
  isRight(X, Y, L).
%Предикат поиска соседних элементов
near(X, Y, L):-
  isRight(X, Y, L);
  isRight(Y, X, L).
%Предикат добавления в список
add(X, [X|_]).
add(X, [_|L]):-
  add(X,L).
```

Стоит также отметить, что множества алгоритмически решаемых задач для разных алгоритмических моделей на сегодняшний день совпадают. Если задача является алгоритмически решаемой, то она решаема в любой алгоритмической модели. То есть если задача решаема в императивной форме, то она непременно решаема и в декларативной форме. Однако удобство различных подходов к решению для различных парадигм, конечно, различно.

Таким образом, логическое программирование – удобный инструмент, который позволяет успешно решать круг задач, связанных с искусственным интеллектом, поиском решений и даже доказательством теорем. Знакомство с одним из представителей данной парадигмы – Prolog-ом будет полезно для программистов любой квалификации для расширения взглядов на программирование в целом и расширения арсенала методов решения задач.

Список использованных источников:

1. Сошников, Д. В. Парадигма логического программирования / В. Д. Сошников. – М.:“Вузовская книга”, 2006. – 220 с.
2. Братко, И. Программирование на языке Пролог для искусственного интеллекта / И. Братко. – М.:“Мир”, 1990. – 560 с.