56. ***Основные понятия логического программирования на языке Пролог (факт, правило, предложение, предикат, терм, унификация). Алгоритм поиска решений, используемый в языке Пролог, на примере предикатов работы со списками.***

Правило состоит из головы (предиката) и тела (последовательности предикатов, разделенных запятыми). Голова и тело разделены знаком :- и, подобно каждому выражению Пролога, правило должно заканчиваться точкой. Интуитивный смысл правила состоит в том, что цель, являющаяся головой, будет истинной, если язык Пролог сможет показать, что все выражения (подцели) в теле правила являются истинными. Тело правила может состоять из нескольких предикатов, объединенных конъюнкцией, которая в запросах обозначается запятой.

Правило, определяющее отношение «сын» через отношение «отец», запишется следующим образом:

сын (X, У) :- отец (У, X).

Это означает, что если человек У является для человека X отцом, то X является сыном для У. Здесь X и У – переменные.

Терм. Центральной структурой в Прологе является терм. Существует 4 вида термов: атомы, числа, переменные и составные термы. Атомы и числа иногда группируют вместе и называют простейшими термами. Терм, который не содержит никаких переменных, называется наземным термом.

Атомы – это обычные строки, являющиеся последовательностью букв нижнего и верхнего регистра, цифр и подчеркивания, начинающиеся со строчной буквы, любые последовательности из возможных символов, заключенные в одинарные кавычки, строки, составленные только из специальных символов: + - * = < > : & (следует отметить, что набор этих символов может отличаться в различных версиях Пролога).

Представленные далее последовательности являются корректными атомами:

B, abcXYZ, x_123, efg_hij, +, ::, < ---- >, ***

Числа в Прологе бывают целыми (Integer) и вещественными (Float).

Не все целые числа могут быть представлены в машине, поэтому их диапазон ограничен интервалом определенным конкретной реализацией Пролога. Обычно реализация допускает диапазон хотя бы от –16383 до 16383, а часто, и значительно более широкий.

Синтаксис вещественных чисел зависит от реализации: 3.14, -0.0035, 100.2. При обычном программировании на Прологе вещественные числа используются редко, т.к. язык Пролог предназначен в первую очередь для обработки символьной, а не числовой информации. При символьной обработке часто используются целые числа, нужда же в вещественных числах невелика.

Переменными в прологи являются строки символов, цифр и символа подчеркивания ‘_’ начинающихся с заглавной буквы или символа подчеркивания, единственный символ подчеркивания является особым случаем и называется анонимной переменной (т. е. Переменной без имени). Она используется, когда значение переменной не имеет никакого специфического интереса. Возможное неоднократное использование безымянной переменной в одном выражении применяется для того, чтобы подчеркнуть наличие переменных при отсутствии их специфической значимости для программы.

X, _4711, X_1_2, Результат, _x23, Объект 2, _

Составные термы состоят из функционального элемента (функтор атома Пролога) и множества параметров (термов Пролога, т.е. атомов, чисел, переменных или других составных термов), заключенных в круглые скобки и отделенных запятыми.

f (g (X, _), 7), ‘My functor’ (dog)

Нельзя помещать символ пробела между функтором и открывающимися круглыми скобками.

Группы составных термов и атомов вместе формируют набор предикатов Пролога.

Предикат – это конструкция вида <имя>(<аргументы>). Аргументы перечисляются через запятую и представляют собой какие-то объекты или свойства объектов, а имя предиката обозначает связь или отношения между аргументами.

Факты – это утверждения о том, что соблюдается некоторое конкретное отношение. Они являются всегда, безусловно, верными. На Прологе факт записывается в виде предиката, аргументы которого являются символьными или числовыми константами. Совокупность фактов – БД.

Факт «Коля работает слесарем» на Прологе запишется следующим образом:

профессия(коля, слесарь).

Вопрос – конструкция ‘?-’ вопросы к БД. Parent(tom, bob).

?-parent(X,bob).

X=tom

YES

Предложение. Предложения могут быть фактами, правилами или вопросами. Все предложения состоят из термов.

Говорят, что два терма унифицируются, если они абсолютно идентичны, или если могут быть сделаны идентичными в смысле совпадения их величин. Совпадение величин означает закрепление за ними некоторого фиксированного значения. Две свободные переменные также унифицируются, потому что они могут конкретизироваться, одним и тем же значением терма. Важно заметить, что такая переменная будет конкретизирована этим значением во всех выражениях. Единственным исключением из этого правила являются анонимные переменные, которые считаются уникальными везде, где они встречаются. ‘/=’ true, если не унифицируются.

Термы больше(X, собака) и больше(осел, собака) унифицируются, потому что переменная X может быть конкретизирована атомом осел. Мы можем проверить это в интерпретаторе Пролога, задав соответствующий запрос, в ответ на который Пролог напечатает также и соответствующее переменной значение:

?- больше(X, собака)=больше(осел, собака).

X = осел

Да

Списки – одна из наиболее часто употребляемых структур в Прологе. Списки заключаются в квадратные скобки и содержат элементы, разделенные запятыми. Элементами списка могут быть любые термы Пролога, т. е. Атомы, числа, переменные и составные термы. Что дает возможность составлять списки из списков. Пустой список записывается как [ ].

[слон, лошадь, обезьяна, собака] – список из 4х атомов

[предок (X, том), [a, b, c], f(22)] – список из 3х составных термов

Первый элемент списка – ***голова***, остальная часть списка – ***хвост***. Пустой список не имеет головы. Список , состоящий из одного элемента, имеет голову и пустой список в качестве хвоста. Разновидностью записи списков является указание головы и хвоста списка. Это делается с использованием символа вертикальной черты. Если этот символ помещается перед последним термом списка, то это означает, что этот последний терм определяет другой список.

?-[1, 2, 3, 4, 5] = [Head | Tail].

Head = 1

Tail = [2, 3, 4, 5]

Yes

Алгоритм поиска решений

Поиск представляет собой просмотр списка на предмет выявления соответствия между элементом данных (объектом поиска) и элементом просматриваемого списка. Если такое соответствие найдено, то поиск заканчивается успехом. В противном случае неуспехом. Результат поиска, так же как и результат любой другой операции Пролога, базирующийся на унификации термов, всегда бывает либо успехом, либо неуспехом. Для сопоставления объекта поиска с элементами просматриваемого списка необходим предикат, объектами которого и являются эти объект поиска и список:

find_it(3,[1, 2, 3, 4, 5]).

Первый из объектов утверждения, 3, есть объект поиска. Второй это список [1,2,3,4,5]. Для выделения элемента из списка и сравнения его с объектом поиска можно применить метод разделения списка на голову и хвост. Стратегия поиска при этом будет состоять в рекурсивном выделении головы списка и сравнении его с элементом поиска. Также как в программе «Голова - хвост» при рекурсии хвостом каждый раз становиться новый список, голова которого присваивается переменной, сравниваемой с объектом поиска. Правило поиска может сравнить объект поиска и голову текущего списка. Саму операцию сравнивания можно записать в виде:

find_it(Head,[Head|_]).

Этот вариант правила предполагает наличия соответствия между объектом поиска и головой списка, хвост списка при этом присваивается анонимной переменной. В данном случае, поскольку осуществляется попытка найти соответствия между объектом поиска и головой списка, то нет необходимости заботиться о том, что происходит с хвостом. Утверждение удовлетворяется в случае совпадения объекта поиска и головы списка. Если нет, то происходит откат и поиск другого правила или факта, с которыми можно снова попытаться найти соответствие.

Для случая несовпадения объекта поиска и головы списка необходимо предусмотреть правило, которое выделило бы из списка следующий по порядку элемент и сделало бы его доступным для сравнения. Поскольку следующий за головой текущего списка элемент является головой текущего хвоста, мы можем представить этот текущий хвост как новый список, голову которого можно сравнить с объектом поиска:

find_it(Head,[Head|_]).

find_it(Head,[_,Tail]):- find_it(Head,Tail).

Если правило find_it(Head,[Head|_]). Неуспешно, то происходит откат, и делается попытка со вторым вариантом find_it.

На втором вхождении предиката find_it Пролог унифицирует имеющиеся термы с заголовком правила find_it(Head,[_,Rest]). При этом первый элемент списка ставится в соответствие анонимной переменной, т.к. оно не представляет интереса: данное правило не было бы задействовано, если бы этот элемент совпадал с объектом поиска при попытке с find_it(Head,[Head|_]).

Теперь нужно присвоить переменной хвост списка, чтобы Пролог попытался установить соответствие между объектом поиска и головой списка хвоста. Попытка удовлетворить рекурсивное правило find_it(Head,Rest). Заставляет Пролог представить хвост текущего как новый самостоятельный список. Опять присвоенный переменной Rest список разделяется на голову и хвост при посредстве утверждения find_it(Head,[Head|_]).

Процесс повторяется до тех пор, пока это утверждение дает либо успех в случае установления соответствия на очередной рекурсии, либо неуспех в случае исчерпания списка.

find_it(Head,[Head|_]).

find_it(Head,[_,Tail]):- find_it(Head,Tail).

Если задать цель find_it(3,[1,2,3,4,5]), то первый вариант правила пытается установить соответствие между головой списка, 1, и объектом поиска,3. Вследствие неравенства 1 и 3 результатом применения этого правила является неуспех. Процесс установления соответствия продолжается со следующей головой списка (уже усеченного), 2, и снова неуспешно. При следующей попытке голова списка, а вместе с ней и объект поиска, равны 3 успешное завершение процесса. На экране появляется True, что как раз указывает на успешное завершение процесса установления соответствия, т.е. на присутствие числа 3 в списке.