Язык SWI Prolog

Обработка списков в программах на языке Пролог

Определение списка

В языках, предназначенных для программирования задач искусственного интеллекта, важную роль играют динамические структуры данных, называемые списками. Главное достоинство списков состоит в том, что при их обработке операции вставки, замены и удаления элементов выполняются весьма просто.

Списки —структуры данных с последовательным доступом, и поэтому основным средством их обработки является рекурсия.

Обозначение списка

Список в языке Пролог представляет собой заключенную в квадратные скобки [ ] последовательность термов, разделенных запятыми:
[<терм1>,<терм2>,…<термn>].

Список — это составной терм или структура, главный функтор которой обозначается символом «.» (точка).

Точечная пара

Структура “Список” может быть представлена в виде так называемой точечной пары •(H,T), где H — первый элемент списка, называемый головой списка, а T —все остальные элементы списка, называемые хвостом списка. Голова списка является термом, а хвост —списком.

В языке Пролог список, разделенный на головы и хвост, обозначается следующим образом: [H|T], например [a|[d,b,k,l]]. Возможность разделения списка на голову и хвост играет важную роль в программировании рекурсивных процедур обработки списков.

Длина списка

Число элементов в списке называется **длиной списка**. Длина списка может динамически изменяться в процессе выполнения запроса. Список, не имеющий ни одного элемента, обозначается [ ] и называется **пустым списком**. Длина пустого списка равно нулю. Пустой список нельзя разделить на голову и хвост.

Представление списка в программе

Голова и хвост списка могут быть заполнены константами, числами или символьными термами, например [1|[6,8,12,9]] или [a|[r,t,y,u]].
Голова и хвост списка могут быть переменными, Например,[H|LT], здесь H — переменная, значение которой является термом, а LT — переменная, имеющая значение списка.

Унификация списков

Списки представляют собой частный случай составных термов, и их унификация выполняется по общим правилам унификации составных термов.

Списки успешно унифицируются, если они имеют одинаковую длину и соответствующие элементы этих списков попарно успешно сопоставимы.

Примеры унификации списков

1) ? - [a]=[].
No

Списки имеют разную длину, унификация невозможна, так как [a] является списком из одного элемента, а [] —пустой список.

2) ? - [L]=[a].
L=a
yes

Списки имеют одинаковую длину, переменная L и терм a успешно унифицируются, и переменная L конкретизируется значением a.

Примеры унификации списков

? - [L]=[a, b, c].
No

Списки имеют разную длину, переменная L обозначает один элемент списка, а другой операнд операции сопоставления является списком из трех элементов.

? - [a|L]=[a, b, c].
L=[b,c]

Yes

Унификация успешна, списки имеют одинаковые первые элементы, переменная L, обозначающая хвост первого списка, успешно сопоставляется и конкретизируется значением [b,c], списком, который является хвостом второго списка.

Типовые процедуры обработки списков

Типовые процедуры обработки списков не являются стандартными предикатами системы программирования языка Пролог.

Предикат length

Предикат определения длины списка length(L,N) является предопределенным предикатом во многих системах программирования на языке Пролог. Схема отношения этого предиката имеет вид: length(<список>,<длина списка>).

Декларативное описание предиката length

Декларативное описание предиката length(L,N) формулируется следующим образом:

Предикат length(X, 0) будет истинным, если X —пустой список, т.е. длина пустого списка равна нулю. Если список можно разделить на голову и хвост, то длина списка равна длине хвоста списка плюс 1.

Процедура length(L,N) состоит из двух правил:

length([ ],0).

length([_|T],N): —length(T,N1),N is N1+1.

Предикат member

Предикат member(X,Y) определяет, принадлежит ли терм X списку Y. Схема отношения этого предиката имеет вид:

member(<терм>,<список>).

Декларативное описание предиката member

Декларативное описание предиката member формулируется следующим образом:

Любой терм X принадлежит списку Y, если терм X является головой списка Y или X принадлежит хвосту списка Y. В этих двух случаях значение предиката member(X,Y) будет истинным.

Процедура member(X,Y) состоит из двух правил:

member(X,[X|_]).

member(X,[_|T]): —member(X,T).

Предикат first

Предикат first(X,Y) определяет, является ли терм X первым элементом списка Y. Схема отношения этого предиката имеет вид:
first(<терм>,<список>).

Декларативное описание предиката first

Декларативное описание предиката first

Формулируется следующим образом:

Терм X является головой списка Y, если терм X сопоставим с первым элементом списка Y.

Процедура first(X,Y) состоит из факта:

first(X,[X|_]).

Предикат last

Предикат last(X,Y) определяет, является ли терм X последним элементом списка Y. Схема отношения этого предиката имеет
вид:
last(<терм>,<список>).

Декларативное описание предиката last

Декларативное описание предиката last формулируется следующим образом:

Если список Y включает только один элемент, и этот элемент сопоставим с термом X, то предикат last(X,Y) истинен. Если список Y можно разделить на голову и непустой хвост и терм X является последним элементом хвоста списка Y, то предикат last(X,Y) истинен. Если терм X отсутствует в списке Y, то значение предиката last(X,Y) —ложь.

Декларативное описание предиката last

Процедура last(X,Y) состоит из двух правил:
last(X,[Y|[ ]]): —X=Y.
last(X,[Z|T]): —last(X,T).

Предикат next

Предикат next(X,Y,L) принимает значение “истина”, если терм X непосредственно следует за термом Y в списке L. Схема
отношения этого предиката имеет вид:
next(<терм>,<терм>,<список>).

Декларативное описание предиката next

Декларативное описание предиката next

формулируется следующим образом:

Если в списке L первые два элемента —X и Y, то предикат next(X,Y,L) истинен, иначе предикат next(X,Y,L) истинен, если термы X и Y являются первыми элементами хвоста списка L.

Декларативное описание предиката next

Процедура next(X,Y,L) состоит из двух правил:

next(X,Y,[X,Y|L]).

next(X,Y,[U|L]): —next(X,Y,L).

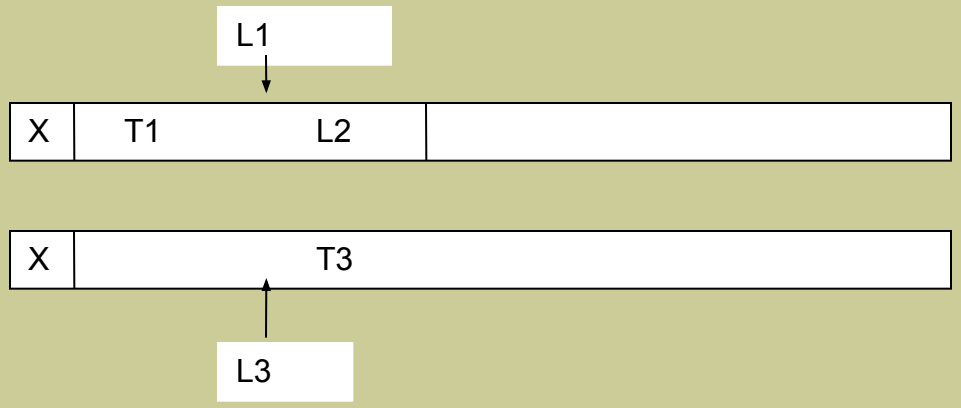
Предикат append

Предикат append(L1,L2,L3) принимает значение “истина”, если список L3 получается путем приписывания все элементов списка L2 после последнего элемента списка L1, как показано на рис. 6. Схема отношения этого предиката имеет вид:
append(<список>,<список,<список>).

Декларативное описание предиката append

Декларативное описание предиката append формулируется следующим образом:

При присоединении к пустому списку любого списка результирующий список будет идентичен присоединенному списку. Если первый список не пуст, то он разделяется на голову X и хвост T1, а результирующий список L3 =[X|T3] , где T3 результат присоединения списка L2 к списку T1.



Декларативное описание предиката append

Процедура append(L1,L2,L3) состоит из двух правил:

append([],L,L).

append([X|T1],L2,[X|T3]): — append(T1,L2,T3).