Практическое занятие «Пролог–5»

08 октября 2019 года

1. Напишите предикат `check/3(+Elem1,+Elem2,+CompFunc)`, принимающий два элемента и имя бинарного предиката. Предикат `check` считается истинным, если является истинным предикат `CompFunc(Elem1,Elem2)`. С его использованием напишите предикат `choosePairs/4(+List1,+List2,+Pred,?ListRes)`, принимающий два списка одинаковой длины `List1` $= [a_1, a_2, \ldots, a_n]$ и `List2` $= [b_1, b_2, \ldots, b_n]$, имя бинарного предиката `Pred` и истинный, если список `ListRes` содержит в каком-то порядке все пары a_i-b_i соответствующих элементов исходных списков, удовлетворяющих предикату `Pred`. (То есть предикат должен корректно работать как в режиме проверки, так и в режиме поиска по последнему аргументу.) Например,

   `choosePairs([1,2,3],[10,2,0],>=,Res)` → `Res = [2-2,3-0]`

   `choosePairs([1,2,3],[10,2,0],>=,[3-0,2-2])` → `yes`

2. Напишите предикат `map/4(+List1,+List2,+MapFunc,?ListRes)`, принимающий два списка и тернарный *вычисляющий* предикат, то есть предикат вида `pred/3(+A,+B,?C)`, истинный, если `C` есть результат вычисления бинарной операции над операндами `A` и `B`. Например, `add(A,B,C) :- C is A+B.` или `append(A,B,C)`. Предикат `map` считается истинным, если список `ListRes` получен из первых двух применением предиката `MapFunc` к парам соответствующих элементов списков `List1` и `List2`; результаты должны быть перечислены в том же порядке, что и в исходных списках. Списки `List1` и `List2` могут быть разной длины; в этом случае, работа ведется по более короткому из них. Например,

   `map([1,2,3], [4,5], add, [5,7])` → `yes`

   `map([[1,2],[],[3]], [[a],[b,c],[d,e,f]], append, Res)` →

   → `Res = [[1,2,a],[b,c],[3,d,e,f]]`

3. Напишите предикат `foldl(X0,Func,Lst,Res)`. Здесь `Func/3(+A,+B,?C)` — тернарный вычисляющий предикат — и `Lst` $= [a_1, a_2, \ldots, a_n]$ — список объектов того типа, которые обрабатываются предикатом `Func`. Предикат `foldl` истинен, если `Res` сопоставляется с результатом цепочки вычислений

   `Func(X0,` a_1`,` x_1`),` `Func(`x_1`,` a_2`,` x_2`),` `...,`

   `Func(`x_{n-2}`,` a_{n-1}`,` x_{n-1}`),` `Func(`x_{n-1}`,` a_n`,` `Res).`

   Например, если имеется предикат `add(A,B,C) :- C is A+B.`, то

   `foldl(0,add,[1,2,3,4],10)` → `yes`

   `foldl(0,add,[1,2,3,4],Res)` → `Res = 10` .

4. Напишите предикат `addHead/3(?Elem,?List1,?List2)`, который принимает элемент и два списка списков и является истинным, если элементы `List2` суть элементы `List1` с приписанным в начале элементом `Elem`. Например, истинны запросы

   `addHead(1, [[],[2],[3]], [[1],[1,2],[1,3]]).`

   `addHead(X, [[],[2],[3]], [[1],[1,2],[1,3]]).  X = 1`

   `addHead(1, [[],[2],[3]], Res).  Res = [[1],[1,2],[1,3]]`

   `addHead(1, Orig, [[1],[1,2],[1,3]]).  Orig = [[],[2],[3]]`

5. Напишите предикат `subsets/2(?L1,?L2)`, принимающий два списка и истинный, если второй список содержит ***в каком-либо порядке*** полный набор подмножеств первого списка, представленных в виде списков. При этом разумно использовать решение задачи 4. Например, истинны запросы

   `subsets([1,2], [[],[1],[2],[1,2]]).`

(элементы списка во втором аргументе могут быть перечислены и в другом порядке)
`subsets([1,2], [[2],[],[1,2],[1]]).`
`subsets([1,2], Res).  Res = [[],[1],[2],[1,2]`
(элементы `Res` могут идти и в другом порядке)
`subsets(Orig, [[1],[],[2],[2,1]]).  Orig = [1,2]` (или `Orig = [2,1]`)

То есть предикат должен корректно работать, как в режиме проверки, так и режиме поиска; причём поиск может вестись как по первому, так и по второму аргументу. Работа в режиме, когда оба аргумента свободны, не предусмотрена, но разумно сообщать об ошибке в этом случае.

6*. *Задача о рюкзаке* имеет следующий смысл. Имеется n предметов, про каждый из которых известен его вес и стоимость. Имеется рюкзак с максимальной грузоподъёмностью w. Какова максимальная стоимость вещей из имеющихся, которые можно унести в таком рюкзаке?

Пусть в базе данных определены предикаты `thing(Name,Weight,Price)`, каждый из которых описывает одну вещь из имеющегося набора. Также пусть определён предикат `sack(MaxWeight)`, задающий максимальную грузоподъёмность имеющегося рюкзака. Напишите предикат `pack(Things,MaxPrice)` истинный, если `Things` есть отсортированный по возрастанию список имён вещей, составляющих какой-либо оптимальный набор, а `MaxPrice` — стоимость этого набора.

Например, при базе
```
thing(plate,1,1.5).
thing(jar,3,5).
thing(silverSpoon,0.3,15).
sack(3).
```
запрос
`pack(Things,MaxPrice).`
должен доказаться и сопоставить переменные следующим образом:
```
Things = [plate,silverSpoon]
MaxPrice = 16.5
```

Примечание: На данный момент неизвестны решения данной задачи, принципиально отличающиеся от полного перебора, поэтому следует реализовать именно полный перебор всех возможных наборов вещей. При этом разумно использовать решение задачи 5.