2.15 Пример декларативного создания программы

Задача об обезьяне и банане

Возле двери комнаты стоит обезьяна. В середине комнаты к потолку подвешен банан. Обезьяна голодна и хочет съесть банан, но не может до него дотянуться, находясь на полу. Около окна этой комнаты находиться ящик, которым обезьяна может воспользоваться. Обезьяна может предпринимать следующие действия: ходить по полу, залазить на ящик, двигать ящик (если обезьяна находится возле ящика), схватить банан (если обезьяна находится на ящике под бананом). Может ли обезьяна добраться до банана?

Мир обезьяны находится в некотором состоянии, которое может меняться со временем.

Состояние обезьяньего мира (объединим функтором):

- вертикальная позиция;
- горизонтальная позиция;
- позиция ящика;
- наличие банана.

Например, начальное состояние обезьяньего мира:

состояние(удвери, наполу, уокна, нет).

Поставленную задачу можно рассматривать как игру для одного игрока – обезьяны. У игры должно быть начальное состояние, цель и правила - ходы.

Начальное состояние:

состояние(удвери, наполу, уокна, нет).

Цель:

состояние(_,_,_, да).

Действия обезьяны:

- перейти (в другое место);
- подвинуть (ящик);
- залезть (на ящик);
- схватить (банан).

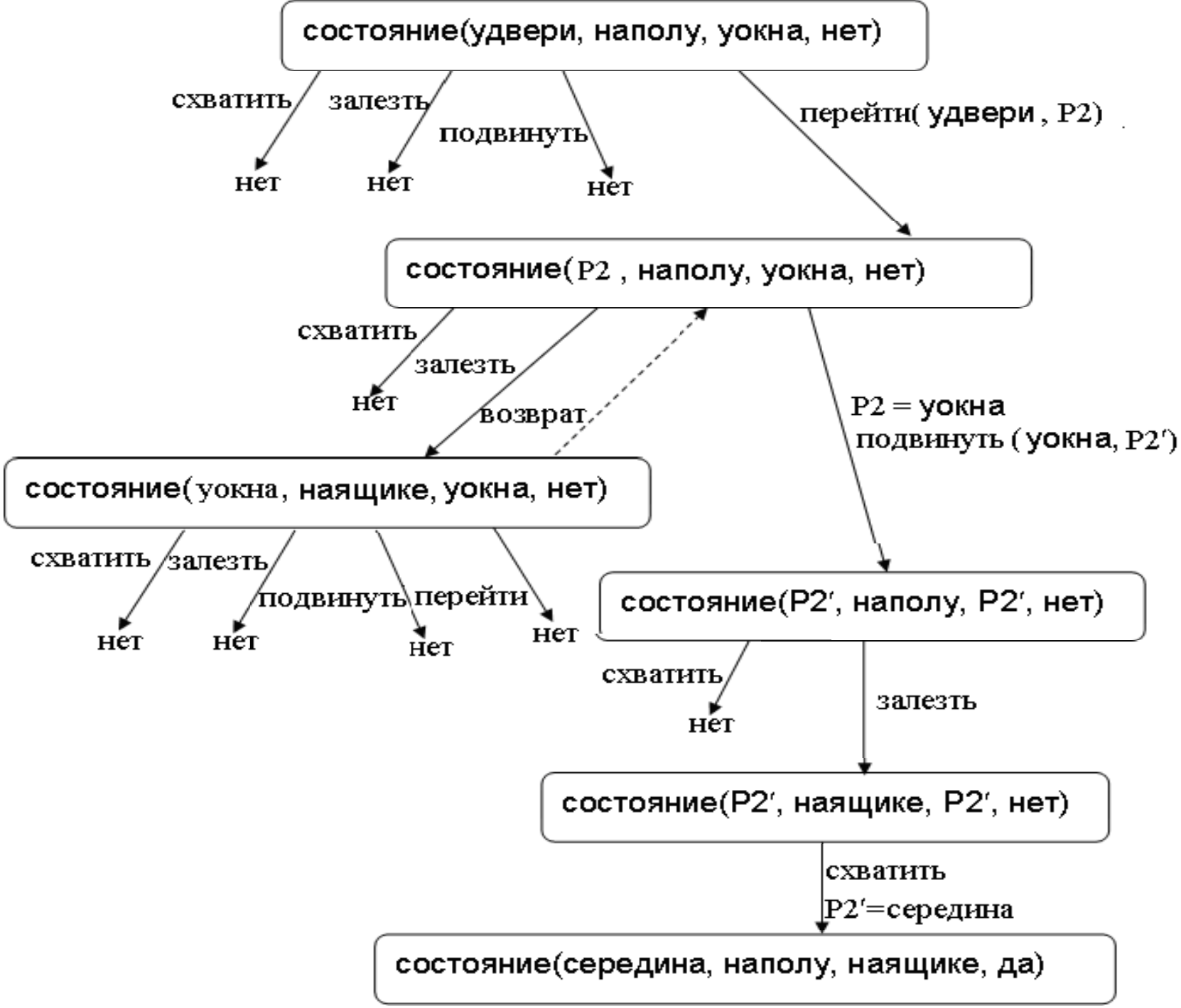
Например, действие «схватить банан» допустимо только для состояния

состояние(середина, наящике, середина, нет)

В результате хода система переходит из одного состояния в другое. Отражение этих переходов - предикат

ход(состояние, действие, состояние).

```
goal:- может_достать(состояние(удвери, наполу, уокна, нет)),!.
ход(состояние(середина, наящике, середина, нет),
      схватить,
      состояние(середина, наящике, середина,да)).
ход(состояние(P, наполу, P, H),
      залезть,
      состояние(P, наящике, P, H)).
ход(состояние(P1, наполу, P1, H),
      подвинуть(P1, P2),
      состояние(P2, наполу, P2, H)).
ход(состояние(P1, наполу, P, H),
      перейти(P1, P2),
      состояние(P2, наполу, P, H)).
может_достать(состояние(_,_,_,да)).
может_достать(S1):-ход(S1, _, S2),
                   может_достать(S2).
```


состояние(удвери, наполу, уокна, нет)
схватить
залезть
подвинуть
перейти(удвери , P2)
нет
нет
нет
состояние(P2 , наполу, уокна, нет)
схватить
залезть
возврат
нет
P2 = уокна
подвинуть (уокна, P2′)
состояние(уокна , наящике, уокна, нет)
схватить
залезть
подвинуть
перейти
нет
нет
нет
нет
состояние(P2′, наполу, P2′, нет)
схватить
нет
залезть
состояние(P2′, наящике, P2′, нет)
схватить
P2′=середина
состояние(середина, наполу, наящике, да)

Для ответа на вопрос сделан только один возврат.
Правильно выбран порядок ходов.

Если поставить первым ходом перейти или подвинуть, то обезьяна будет ходить туда-сюда или двигать ящик в разные стороны.

Важен порядок предложений и целей в программе!

2.16 Решение логических задач с использованием списков

Задача о фермере, волке, козе и капусте

Фермер (**farmer**), волк (**wolf**) , коза (**goat**) и капуста (**cabbidge**) находятся на одном берегу. Всем надо перебраться на другой берег на лодке. Лодка перевозит только двоих. Нельзя оставлять на одном берегу козу и капусту (коза съест капусту), козу и волка (волк съест козу).

Процесс перевозки - последовательность состояний state с четырьмя аргументами, каждый из которых может принимать 2 значения: left или right и отражает соответственно нахождение фермера, волка, козы и капусты.

Например, state(left,right,left,right) отражает, что фермер и коза находятся на левом берегу, а волк и капуста – на правом.

Фермер может перевести на другой берег, кроме себя, либо волка, либо козу, либо капусту.

Для описания перевозки определим предикат move, описывающий переходы из одного состояния в другое. Для переправления с одного берега на другой введем предикат opposite, который определяет сторону берега, противоположную исходной.

Например, перевозка волка на другой берег:

```
move(state(X,X,G,C),state(Y,Y,G,C)):-opposite(X,Y).
```

С помощью предиката danger определим опасные состояния (волк и коза на одном берегу, а фермер на другом или коза и капуста на одном берегу, а фермер на другом).

Предикат path формирует список из последовательности состояний, решающих поставленную задачу.

Будем добавлять в результирующий список новое состояние, если в него возможен переход из текущего состояния, оно не опасное и не содержится в сформированной части списка (для избегания зацикливания).

```
goal:-S=state(left,left,left,left),
        G=state(right,right,right,right),
        path(S,G,[S],L),reverse(L,L1),
        nl,writeln('A solution is:'),
        writeln(L1).
move(state(X,X,G,C),state(Y,Y,G,C)):-opposite(X,Y).
move(state(X,W,X,C),state(Y,W,Y,C)):-opposite(X,Y).
move(state(X,W,G,X),state(Y,W,G,Y)):-opposite(X,Y).
move(state(X,W,G,C),state(Y,W,G,C)):-opposite(X,Y).
opposite(right,left).
opposite(left,right).
danger(state(F,_,X,X)):-opposite(F,X).
danger(state(F,X,X,_)):-opposite(F,X).
path(G,G,T,T):-!.
path(S,G,L,L1):- move(S,S1),
                 not(danger(S1) ),
                 not(member(S1,L)),
                 path(S1,G,[S1|L],L1),!.
```

Решение числовых ребусов

```
  APAPA
+ SLEEPY
--------
 ETUDES
```

Задача состоит в том, чтобы заменить все буквы цифрами. Одинаковым буквам должны соответствовать одинаковые цифры, а разным буквам – разные цифры.

Определим предикат sum(N1,N2,N), который истинен, если существует такая замена букв цифрами в словах N1, N2, N, что N1+N2=N.

Числа будем представлять списком цифр, из которых оно состоит. Будем считать, что все списки имеют одинаковую длину.

Вспомним как выполняется сложение столбиком. Суммирование выполняется справа налево с учетом переноса из предыдущего разряда. Следовательно, нужно хранить информацию о переносе из предыдущего разряда и о переносе в следующий разряд. Поскольку разным буквам должны соответствовать разные цифры, то при суммировании цифр нужна информация о цифрах, доступных до и после сложения (эта информация будет представляться двумя списками).

sum(N1,N2,N, P_before,P_after,
Cifri_before,Cifri_after)

Если у чисел есть хотя бы одна цифра, то суммируем числа без самой левой цифры и, используя список оставшихся цифр и перенос из предыдущего разряда, суммируем самые левые цифры и добавляем полученную сумму в результат суммирования без самой левой цифры.

Для суммирования цифр определим предикат sumc.

sumc(D1,D2,D,B, A,LB,LA)

D1,D2 – цифры

D – цифра результата в этом разряде

B – перенос из предыдущего разряда

A – перенос в следующий разряд

LB – список цифр для выбора до сложения

LA - список цифр для выбора после сложения

Если какая-то цифра не конкретизирована, то ее необходимо конкретизировать какой-нибудь цифрой из списка LB.

Для получения значения неозначенной переменной будем использовать предикат delete.

```
sum(N1,N2,N):-
                sum1(N1,N2,N,0,0,[0,1,2,3,4,5,6,7,8,9],_).
sum1([],[],[],0,0,L,L).
sum1([D1|N1],[D2|N2],[D|N],C1,C,L1,L):-
                sum1(N1,N2,N,C1,C2,L1,L2),
sumc(D1,D2,D,C2,C,L2,L).
sumc(D1,D2,D,C2,C,L2,L):-delete1(D1,L2,L3),
                                delete1(D2,L3,L4),
                                delete1(D,L4,L),
                                S is D1+D2+C2,
                                D is S mod 10,
                                C is S//10.
delete1(X,L,L):-nonvar(X),!.
delete1(X,[X|L],L).
delete1(X,[Y|L],[Y|L1]):-delete1(X,L,L1).
```

```
  БАЙТ
+ БАЙТ
-------
 СЛОВО
```

?- sum([0,Б,А,Й,Т], [0,Б,А,Й,Т], [С,Л,О,В,О]).

Б = 3

А = 6

Й = 4

Т = 1

С = 0

Л = 7

О = 2

В = 8;

…….

false

Этот ребус имеет 14 решений.