Государственное учреждение образования
“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет информационных технологий и управления
Кафедра интеллектуальных информационных технологий

# Лабораторная работа №2

по дисциплине «Логические основы интеллектуальных систем»
на тему «Логическое программирование поиска решения задачи»
**Вариант 2**

Выполнил:
студент гр.121702
Витковская С. И.

Проверил:
Ивашенко В. П.

Минск 2023

**Цель**: Приобрести навыки логического программирования поиска решения задачи.

**Задача:**

Два берега реки. На одном из них – человек, который несет капусту, ведет козу и пойманного волка. Требуется с помощью лодки, вмещающей не более одного животного или предмета, переправиться на другой берег. Человек не может оставлять козу с капустой и волка с козой.

**Дополнительные теоретические сведения:**

Грамматика языка PROLOG.

<ПРОЛОГ-предложение> ::=<правило> | <факт> | <запрос>
<правило> ::=<заголовок> ‘:-’<тело>
<факт> ::=<заголовок> ‘.’
<запрос> ::=<тело>‘.’
<тело> ::=<цель> /’,’<цель>/’.’
<заголовок>::= <предикат>
<цель>::= <предикат> |<выражение>
<предикат>::= <имя>/ ‘(‘<терм> /’,’<терм>/ ‘)’/
<терм>::= <атом> |<предикат>|<список>
<атом>::= <переменная> |<число> |<строка>|<имя>
<список>::= <список с заголовком>| <простой список>
<список с заголовком >::= ‘[‘ <терм >/’,’<терм>/’|’ < терм>’]’
< простой список>::= ‘[‘ <терм >/’,’<терм>/’]’|‘[’]’
<выражение>::= <терм> /<оператор><терм>/
<оператор>::= ‘is’ | ’=’ | ‘==’ | ’¯’ | ’>=’ | ’=<’ | ‘=¯’ | ‘=:=’

**Правила:**

- state(left, left, left, left, _) – предикат, который представляет собой конечное состояние, которого должна достичь программа. Является граничным условием, которое позволяет остановить рекурсию.

- state(left,Wolf,Goat,Cabbage,_) – предикат, реализующий переплывание реки с левого берега на правый человеком в пустой лодке. Wolf,Goat,Cabbage хранят в себе значения берегов на которых находятся соответствующие объекты.

- state(Ride_together, Wolf ,Ride_together, Cabbage, Last) – предикат, реализующий переплывание реки с одного берега на противоположный человеком в лодке с козой. Wolf, Cabbage хранят в себе значения берегов на которых находятся соответствующие объекты. Ride_together – берег, с которого производится перемещение козы. Last – кого перемещали в прошлый раз.

- state(Ride_together, Ride_together, Goat,Cabbage,Last) – предикат, реализующий переплывание реки с одного берега на противоположный человеком в лодке с волком. Goat,Cabbage хранят в себе значения берегов на которых находятся соответствующие объекты. Ride_together – берег, с которого производится перемещение волка. Last – кого перемещали в прошлый раз.

- state(Ride_together,Wolf,Goat,Ride_together,Last) – предикат, реализующий переплывание реки с одного берега на противоположный человеком в лодке с капустой. Wolf, Goat хранят в себе значения берегов на которых находятся соответствующие объекты. Ride_together – берег, с которого производится перемещение капусты. Last – кого перемещали в прошлый раз.

- dangerous_state(Man, Comparison_object1, Comparison_object2) – предикат, проверяющий, безопасным ли будет состояние, в котором окажутся Comparison_object1 и Comparison_object2, при том или ином перемещении. Comparison_object1 и Comparison_object2 – объекты которые не должны оказываться на одном берегу без присутствия человека.

**Факты:**

opposite(left,right). – если задан левый берег, то противоположный ему будет правым.

opposite(right,left). – если задан правый берег, то противоположный ему будет левым.

**Встроенные предикаты:**

- concat(Str1, Str2, Str3) – конкатенация строк Str1 и Str2 в строку Str3
- not() – логическое отрицание
- write() – вывод на экран

**Замена именований:**

| | |
|---|---|
| state | S |
| dangerous_state | D |
| opposite | O |
| Man | M |
| Wolf | W |

| | |
|---|---|
| state | S |
| dangerous_state | D |
| opposite | O |
| Goat | G |
| Cabbage | C |
| Comparison_object1 | CO1 |
| Comparison_object2 | CO2 |
| Ride_together | RT |
| Other_side | OS |
| Last | L |
| = | is |

**Описание предикатов:**

state(right,right,right,right,nothing)

state(Ride_together,Wolf,Ride_together,Cabbage,Last):-

not(Last=goat),opposite(Ride_together,Other_side),

write('Try drove the goat '),write(Other_side),nl,

not(dangerous_state(Other_side,Other_side, Wolf)),

not(dangerous_state(Other_side,Other_side, Cabbage)),

state(Other_side, Wolf, Other_side,Cabbage,goat),

concat('Drove the goat ',Other_side,String), write(String),nl.

((((!is(L,goat) /\ O(RT,OS)) /\ !D(OS,OS,W)) /\ !D(OS,OS,C)) /\
/\S(OS,W,OS,C,goat)) -> S(RT,W,RT,C,L)

((((is(L,goat) \/!O(RT,OS)) \/  D(OS,OS,W)) \/ D(OS,OS,C)) \/
\/!S(OS,W,OS,C,goat))\/S(RT,W,RT,C,L)

((((is(nothing, goat) \/ O(right, left) \/  D(left, left,right) \/ D(left, left, right) \/
\/!S(left, right, left,right, goat) \/ S(right, right, right, right, nothing)

```
dangerous_state(Man,Comparison_object1, Comparison_object2):-
    Comparison_object1=Comparison_object2,
     not(Comparison_object2=Man),
    write('danger position'),nl,!.
```

(is(CO1,CO2) /\ !is(CO2,M))->D(M,CO1,CO2)

(!is(left,right) \/ is(right,left))\/D(left,left,right)

```
state(left,Wolf,Goat,Cabbage,_):-
    write('Try went empty on the right'),nl,
    not(dangerous_state(right,Goat,Cabbage)),
    not(dangerous_state(right,Wolf,Goat)),
    state(right,Wolf,Goat,Cabbage,nothing),
    write('Went empty on the right'),nl.
```

((!D(right,G,C) /\ !D(right,W,G) /\ S(right,W,G,C,nothing)) -> S(left,W,G,C,_)

((D(right,left,right) \/ D(right,right,left)) \/ !S(right,right,left,right,nothing)) \/ \/S(left,right,left,right,_)

**Дерево вывода:**

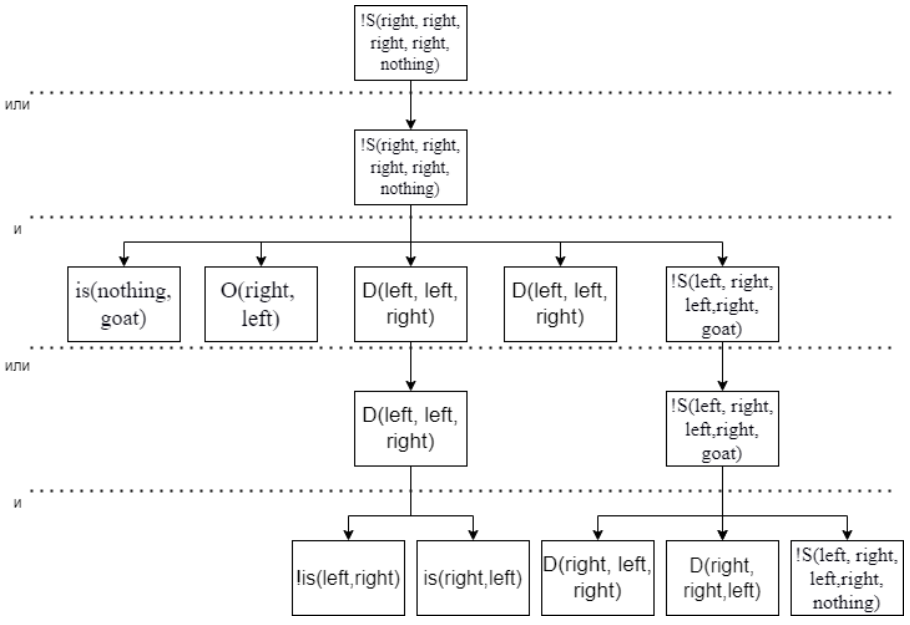


Рис 1. Дерево вывода программы

**Вывод:** Приобрести навыки логического программирования поиска решения задачи. В результате выполнения лабораторной работы была составлена программа поиска решения для задачи перевозки козы, волка и капусты.

Список использованных источников:

1. Логические основы интеллектуальных систем. Практикум : учеб.-метод. пособие / В. В. Голенков [и др.]. – Минск : БГУИР, 2011. – 70 с. : ил. ISBN 978-985-488-487-5.

2. SWI-Prolog Documentation [Электронный ресурс]. – Режим доступа: https://www.swi-prolog.org/pldoc/man?section=libpl. – Дата доступа: 22.05.2023.