1) Consider the structural representation of a combinatorial circuit, as a set of logic gates. A circuit

Combinatorial logic is represented by a shape term

circuit (Gates), or Gates is a list of logic gates or comments. Each door is optionally preceded by a comment. A logic gate is a term of the form Gate (Name, Type, Inputs, Output). name is a unique name associated with the respective door. Type is the type of the door and can be or, and, xor, not, nor or nand. inputs represents the input of the respective door. For doors not, there is only one entrance, which is represented by a symbol. For other doors, inputs is a list of symbols. Output is a symbol which shows the output of the gate.

For example, a full adder for a bit is represented by:

circuit([

Gate (x1, xor, [i1, i2], t1), com ('Gate to generate the sum bit),

gate (x2, xor, [t1, i3], O1), gate (a1, and, [i1 , i2], t2), gate (a2,

and, [i3, t1], t3),

com ('Gate to generate the transmission bit'), gate (O1, gold, [t3, t2],

o2)]).

requirements:

a) Define the predicate circuit (X), that is true if X is a term, which is a combinational logic circuit, in the form of a list of circuits.

b) Set the predicate delete_comments (Ci, Ce), that is true if This is the description of the circuit obtained by eliminating the feedback of the circuit description This.

c) Set the predicate connexions_list (C, Connections) that is true if

connections is the list of all the circuit connections names C.

d) Assume that the values of the input signals of the circuit are given in the form of a list, Signals, with elements of the form signal (input, Value). For example, if the full adder receives as input values i1 = 0,

i2 = 1, i3 = 1, then this list will be [signal (i1, 0), signal (i2, 1) signal (i3, 1)]. To define

the predicate signal (Connection, Circuit,

Inputs, Value) that is true if Value is the connection value Connection

circuit Circuit, for entries with values in the list Inputs.

2) Consider the problem of 15-puzzle, for a square grid size of 4 * 4. The grid contains 15 square slippery. Each square is labeled with a natural number between 1 and 15. There is an empty square which allows the sliding of the square, with a position on the left, right, top or bottom.

It is necessary to find a minimum sequence of movements which transforms an initial configuration in a given final configuration. The problem can be solved using a search strategy in the state space. A state of the problem is given by the configuration of the squares on the grid. This configuration can be represented by a term st (List, Position). list is a list of 16 natural numbers. The numbers represent the label squares, clicking on the lines, from left to right. The open square is considered to be tagged with the value 0. Position represents the index of the empty position in the list (the indexes are numbered from 1). For example, the following table:

1	3	6	9
10	2	13	15
14	7		5
8	4	11	12

is represented by the Prolog list:

st ([1,3,6,9,10,2,13,15,14,7,0,5,8,4,11,12], 11).

requirements:

a) Define the predicate state (X), that is true if X is a term that represents a state of the problem of the 15-puzzle.

b) Set the predicate neighbors (S0, LS1), that is true if LS1 is a list of upcoming states, according to the state S0, the problem of the 15-puzzle. The movements are considered in the following order: left, top, right, bottom.

c) To solve the problem of the 15-puzzle using heuristic search algorithm, it is necessary to use a heuristic function, the value is an estimate of the number of movements needed to move from the current state to the final state. Is S0 and S1 two states. The distance that separates them can be estimated by counting all the squares pairs placed in similar positions in the two states, but labeled with different values. Define the predicate eval_h (S0, S1, H), that is true if H is the value of the heuristic estimate of the distance between the states S0 and S1 the problem of the 15-puzzle.

d) Define the predicate neighbours2 (S0, LS1), that is true if LS1 is the list of successor states, according to the state S0, obtained by performing both movements. The movements are considered in the following order: left, top, right, bottom. Back in S0 will be ignored.

3) Solve a problem version of the monkey and banana. For more details on this issue,

read the file seal and see Monkey.pdf film

https://www.youtube.com/watch?v=T095SaEhHas

In one room, there is a monkey, a box in the corner (or by the window) and a banana hanging from the ceiling in the middle of the room. The monkey wants to eat bananas, but he can only do that if he moves to the box, places the box under where banana is suspended up to the box and jumps to get the banana.

Solving the version of the monkey of the problem and banana, which is extended as follows:

a) The environment has 4 B, labeled 1, 2, 3, 4

b) There are doors of the chamber 4 to the chambers 1, 2 and 3

c) To enter a room and come out, the monkey has to open the door

d) Each room has a window

e) In each room there is a box (labeled as 1, 2, 3, 4, depending on the room is located in the box), near the window

f) The banana is placed in the chamber 4, suspended from the ceiling

g) The monkey may initially be in any room, in one of the following locations: near the door in the middle of the room, near the window

h) The boxes can be stacked in the same room, but in an ascending order of their index (it is possible to put the box 3 above the box 1, the box 4 on top of the box 2, but it is not possible to put the box 3 above the box 4)

i) Because bananas are suspended at a high height, monkey must place the boxes 1, 2, 3 and 4 one above the other (in this order) in the chamber 4, reaching banana

The program should display the solution found in the form of a sequence of states in which the monkey is randomly placed in one of the rooms.