Introduction to Artificial Intelligence
Laboratory work

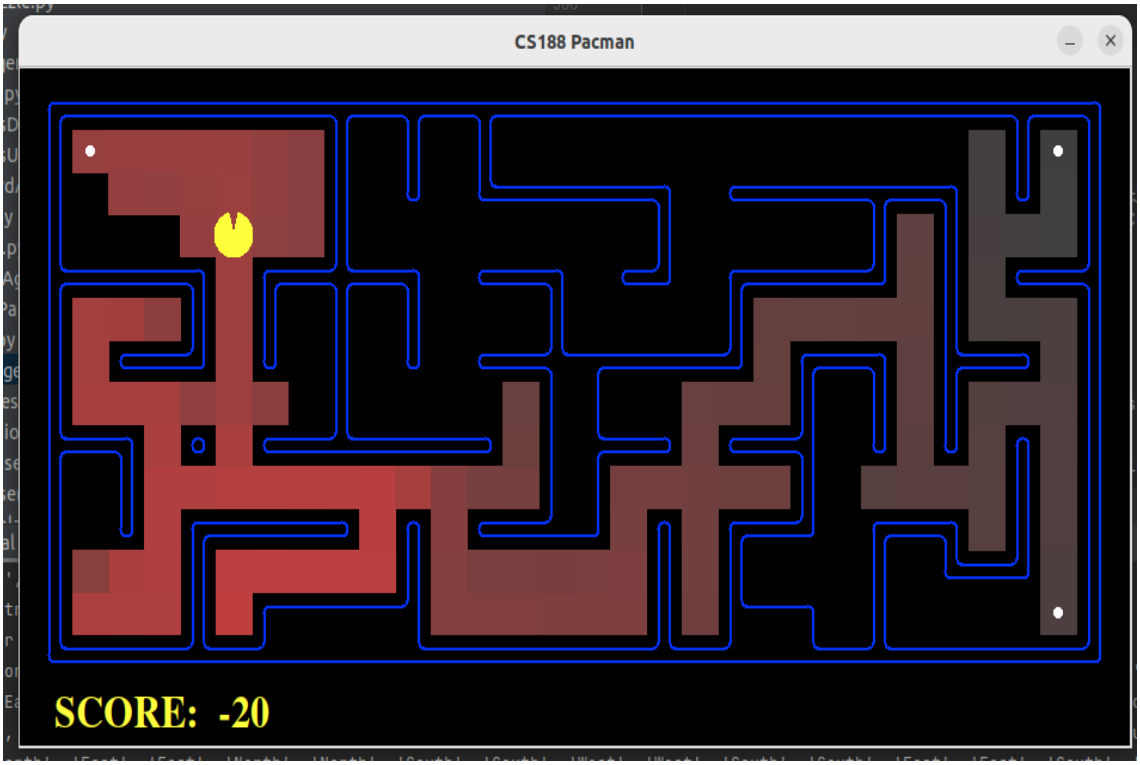
Andercou Alexandru

December 9, 2022

Laborant Aron Katona
Assoc. Prof. eng. Adrian Groza

Contents

1 A1: Search



1.1 Introduction

Pacman is an arcade game with a long history. The main goal of the Pacman is usually to eat as much food as possible avoiding threat represented by ghosts and walls. However a sub-task that can be attributed to a Pacman is to find the special dots that transform ghosts in food, these special dots are in number of 4 and are located always in the corners. So the problem of finding the special dots turns into the https://www.overleaf.com/project/6366ae9e717c1ba80500f282Corners Problem.

1.2 Defining the problem and conceptual solutions

In order to solve the corners problem ,which means finding the shortest path that connects the Pacman and the 4 corners there exists many local search algorithms from the most simple like breath first search and depth first search to A*,weighted A* star ,and many other variants. As the scope of this project I choose to explore 4 variants of weighted A* with 5 heuristics starting from the simplest to more complex ones.

A* is a type of a greedy local search algorithm that uses a cost function with formula f=g+h where g represents the total cost till the current state and h represents the cost of the heuristic

Weighted A* is a simple variant of A* which complicates a bit the formula used by A*. It uses for the cost the function f=cg*g +ch*h where cg and ch are the weights attributed to the total cost and to the heuristic and it can be translated into the importance the agent offers to the total cost and to the knowledge or to his estimation, by varying the value of cg and ch we get a vast variaty of agents that act differently.If we set ch to 0 than we get a simple greedy algorithm that considers only the total cost till the current state, this variant gets outside of A* and weighted A* domains so it will not be discussed.

1.3 Corners Problem Heuristics

In order to get an estimation that is closest to the optimal one the heuristics must be consistent and admissible so it must not be bigger than the actual real optimal cost and it must respect the triangle law. I come up with 4 heuristics.

1.3.1 Heuristic 1

The most simple heuristic you can think of is the number of corners the agent managed to get to. The first heuristic is decreasing each time the Pacman manages to get in a corner.It will almost always be smaller than the total real cost, but it is a very pour estimator overall because most of

1.3.2 Heuristic 2

Another slightly smarter method is estimating the rest of the path by the biggest distance towards a corner using Manhattan distance that estimates better the cost of walking in a grid structure than euclidean distance. This heuristic assumes no walls.We hope that once it gets to the furthest corner it will already get trough all the other corners first.

1.3.3 Heuristic 3

For the 3th heuristic we make a few changes to the 2th heuristic. For starters besides the distance from Pacman to a corner we take in consideration also the distance between any two corners distance that Pacman we will anyway have to pass trough. Another important change is that we replace Manhattan distance with mazeDistance which will estimate the distance between corners using simulation with a bfs algorithm.We will consider a sum between the smallest distance between Pacman and a corner and the biggest distance between any 2 corners.

However bfs is an expansive algorithm and with not work well for a big maze,especially that we need to compute it many-many times.

1.3.4 Heuristic 4

As the 4th heuristic instead of considering the maximum distance between corners we will be more positive and consider the average of the distances between corners, in the rest the method stays the same.This modification brings us more to reality , there might be beteer paths than taking the longest path between corners.

1.3.5 Heuristic 5

For the fifth heuristic we combine the 2th , 3th and 4th. For this heuristic we will drop mazeDistance because it is too expensive and not fiting for most big problems and use a formal distance Manhattan distance. From the 3th heuristic we will keep combining the distance between corners and distance between Pacman and a corner but from the 4th heuristic we will use the average of distances between corners to not overevaluate the real distance.

1.4 Weighted A* variants

In weighted A* algorithm by modifying the weights associated with total cost and heursitic we get multiple possibilities. I decided to test 5 of them

1.4.1 Weighted A* with cg weight bigger than than ch and none 0

This method translated to a conservative Pacman but slighly open to liberalism or to free thinking , to estimations of reality.

1.4.2 Weighted A* with cg weight smaller than ch and none 0

This translates to a liberal who relies on estimations but still listens to the pragmatism of real cost, as it can't completly avoid reality.

1.4.3 A*

A* is the Variant of Weighted A* where both of weights have the same value. It will be used as a control method , to see the difference caused by algorithms. It translated to a balanced person who values both reality and estimations.

1.4.4 Greedy on heuristic without cost weight

If the weight of total cost is set to 0 we get an agent that considers only the heuristic , it is likely to not get great results as many positions will have the same heuristic. It represents an impulsive agent

1.4.5 Dynamic Weighted A*

In the previous cases we considered A* with fixed weights, if we allow change of weights as the search progresses we get a different result from any other ,likely a combination between the other behaviours.This one is more akin to a real agent as humans are able to change their method of action as they progress in life. For this heuristic we will consider the cg as 1 and ch=1+epsilon-(epsilon*depth(n))/PathEstimation Where path estimation is the lenght of path resulted from a bfs traversal. where epsilon is a random number between 0 and 1 and depth is a function that grows with each node that is taken out of the queue.

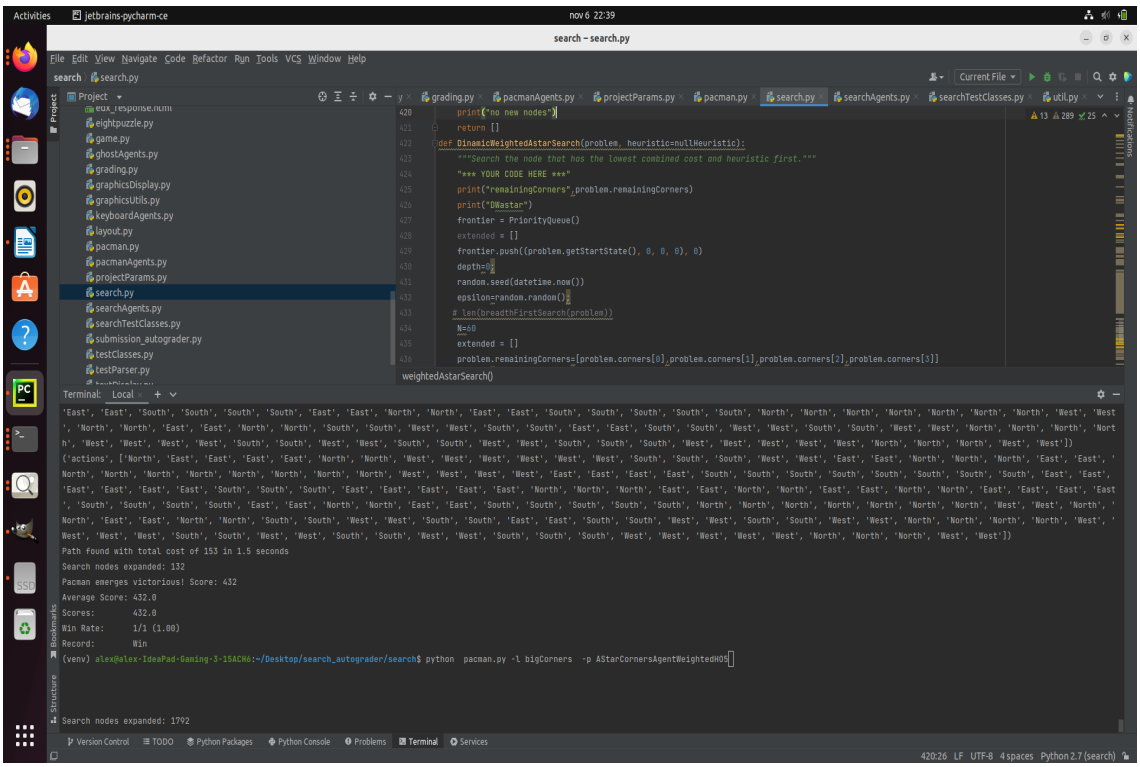
1.5 Implementation and results

We tested all the combinations between one given heuristic and one algorithm of search resulting in 5*4=20 different agents

The testing was made on the grids:tinyCorners and bigCorners.

1.5.1 Testing

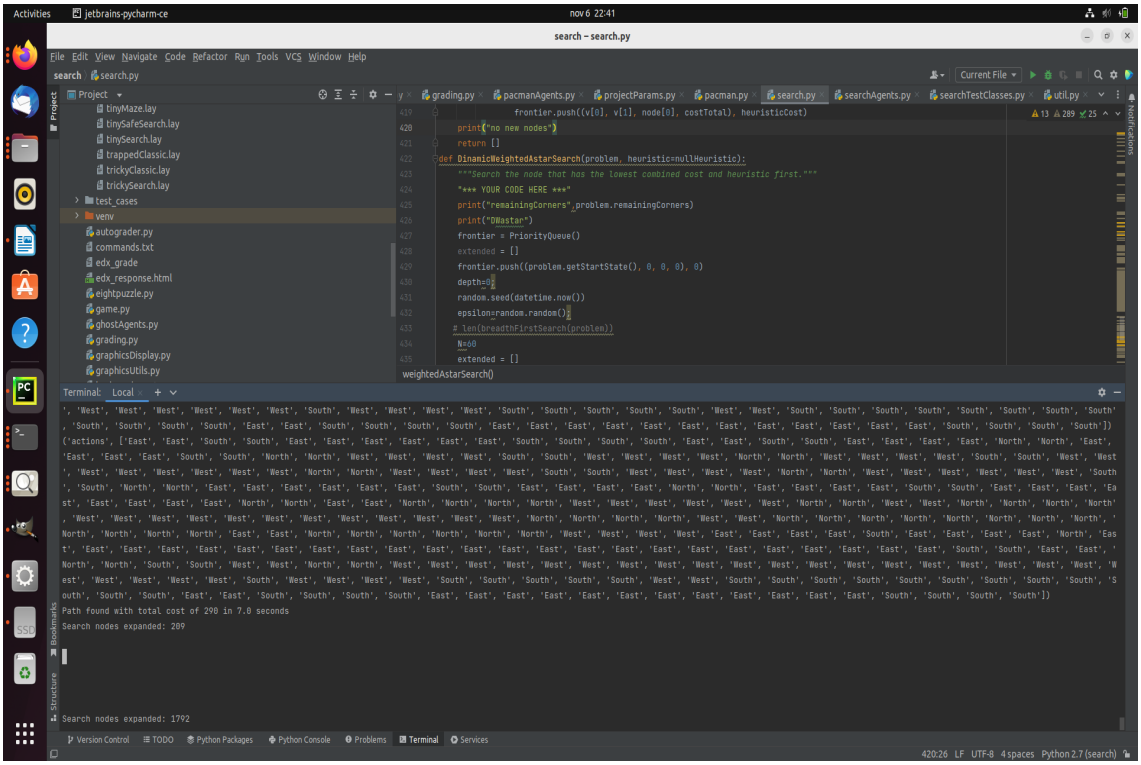
For testing we used PyCharm terminal with the command
$python pacman.py - layOut_name - pAgentTypeClass$
The layOut we used where mediumCorners and bigCorners.



1.5.2 Results

After Aplying the command in terminal for each eoch thec 25 combinations we made a table using the number of Node expanded and running time where was relevant. Where is put in the table medCorn means it was way to slow to be reasonably run on the bigCorners layout.

A/H	H1	H2	H3	H4	H5
WC	1062	812	365 medCorn 67.9s	232 medCorn 31.2s	497
WH	713	880	118 medCorn 44.7s	164 bigCorn 38.3s	209
A*	1062	834	171 medCorn 95.5s	202 bigCorn 41.6s	363
GH	782	1193 29.9s	126 medCorn 45.9s	173 bigCorn 77.1s	192 23.8s
DW	731	704	133 49.6s	252 big Corn 44.9s	751 29.6s



55

SO after the experimentations, if we round the numbers we get that the combination that produced the best results were with GH and WH variants which put a lot of importance on the heuristic but GH proved to be the worst with the 2th heuristic. The 3th and 4th heuristics can be resonably used only on small problems in this case a medium problem , for H3 and H4 the one that performed the best was again WH.

2 A2:Logics

2.1 MineFOL and an Escape Jail game

MineFol is a single person grid game. The purpose of the player is to traverse the grid by avoiding the dangers represented by mines. In the players help , he gets messages in natural language and fol that indicate where are or where are not the mines. The player wins if he correctly avoids all the mines and visits all the cells that do not contained a mine. If the player gets in a cell with a mine he dies and loses the game.

In the Escape Jail game , you are a wrongfully accused prisoner that desperately needs to escape from the grid-like prison. In order to escape you need to find the only key of the prison and the exit.The problem is that nobody knows where they are. Also the prison is not a easy place to escape from. There are a handful of different dangers and you do not know where they are. But no worry, there are other prisoners that know that. If you are able to find friends ,avoid dangers find the key and door you win your right to leave the prison and get to freedom.

2.1.1 FOL

FOL or first order logic is a language used to model problems and prove theorems. It represents object properties ,situations,hypotheses and theorems by using variables,constants,predicates, functions and logical ,existential and universal operators.

Predicates are entities that take as parameters constants or variables and return a value of truth: Truth or False.Functions are as predicates but they return integer values contained in the domain of the specific problem. Logical operator are :

$$\&, \|, -$$

known as and ,or and not, insertedinserted and they are used to connect predicates and variables with values of truth.

Existential operator

$$\exists$$

exists , represents parts of the domain of a variable while the universal operator

$$\forall$$

all represents the entire domain of a variable.

2.1.2 Prover9 and Mace4

Prover9 is a theorem prover. It takes as input clauses ,hypotesis or assumptions and theorems or goals represented in propositional or first order logic and returns a proof if the theorem can be proved or failure if it can't prove the goal. Prover9 uses reduction to abssurd by negating the goals and trying to find inconsistensies with the rest of clauses. If it can find an incosistency a value of fals it returns THEOREM PROVED together with the steps taken to pove the theorem.

Mace4 is a powerful model builder and checker. It takes as input a domain for variables and conditions or assumptions that the models must respect. It tries to build all the models possible or as many models as specified that respect the rules given. Mace4 models functions and predicates that take inputs as tables, and assigns value to variables.Beside building models it can be used to check theorems by contradiction , if mace4 can't find a model for the given domain ,it means that there is an inconsistency in the model ,and it is unjustifiable at least for the domain given , so by negating your goals and adding them to the assumptions you can check if there are situations where your theorem is false that would make the theorem false as a hole.

2.1.3 Fol modeling of MineFOL and Escape Jail game

The games were modeled having two roles with two different knowledge bases : a player and a game master so there are two different setups with different laws written in fol.

The predicates mine,safe,visited,key,door take two arguments: x and y , where x represents the row and y represents the column in the grid and they are both indexed starting from 1. The setup for the player contains the safety rule:

$$mine(x,y) -> -safe(x,y) and mine(x,y) | safe(x,y)$$

and the fact that tells that the first cell is always safe:

$$safe(1,1).$$

The setup for the game master contains the rules regarding the win and lose as well as the predicates mine for locations where there are found mines , -mine for locations that do not contain mines ,and the predicate visited negated for all locations in the grid excepting the cell 1,1.

The rule for losing is the same for both Escape jail game and Mine Fol :

$$existsx existsy(x >= 1 \& y >= 1 \& (visited(x,y) \& mine(x,y))) -> lose.$$

This expression is quite obvious , if there is a position that was visited by the player and that position contained a mine means you died and lost the game.

For winning there are two different laws for MineFol and Escape jail game because they differ in the conditions necessary for winning the game. the predicate that represents winning was chosen to be Win because win with lower case is a variable in mace4 and prover9.

The rule for winning for MineFol is that the player visited all cells that do not contain a mine. In MineFol implementation this rule appears negated: if there exists a position that was not visited and that position does not contain a mine you still didn't win the game:

$$existsx existsy(x >= 1 \& y >= 1 \& -visited(x,y) \& -mine(x,y)) -> -Win.$$

Because it is written in this way the goal when it will be checked by Mace4 will be Win.

The rule for winning in Escape jail game is that you need to find the door and the key to escape the jail:

$$door\_found\&key\_found <-> Win.$$

In this came when checking using Mace4 we need to give the goal for win negated:-Win.

The Escape jail game has 4 more rules than MineFol :

$$existsxexistsy(x >= 1\&y >= 1\&key(x,y)\&visited(x,y)) <-> key\_found$$

,

$$existsx(existsy(x >= 1\&y >= 1\&visited(x,y)\&door(x,y))) <-> door\_found$$

,

$$door(x,y)\&door(z,w) -> x = z\&y = w, key(x,y)\&key(z,w) -> x = z\&y = w.$$

The first two laws tell that the player needs to find the key and door and the second tell that there exists one single door and one single key. This game also contains predicates door(doorx,doory) and key(keyx,keyy) where doorx,doory is the position in grid of the door and keyx and keyy is the position of the door is located.

As the game evolves and current location is changed, the game master knowledge base changes too.At each new location visited a predicate of visited is added and a predicate of -visited is removed for that specific position.

The players knowledge base changes by adding predicates for messages in fol as they are discovered by the player.

2.1.4 Python modeling of MineFOL and Escape Jail game

For the games there is neccesary to save the positions of the messages, the obstacles, and the visited positions for the entire board for this we use the class Propety which contains a matrix with all the positions on the board and for each position a list with 2 elements, the first element tells if the location is empty or not and the second contains the information found at that position. This class is instantiated for fol messages,natural language texts for messages, mines, and visited locations.

Messages,bombs, and other props as : key or door are loaded from files using the methods: load_messages , load_obstacles,load_key_door and they are saved in Propety objects : text,msgs,obst. Key and door are saved in the global variables: keyx,keyy,doorx,doory. For the assumtions of the game masster assumptions_ruler list is used and for the player it will use the default list assumptions.

2.1.5 MineFOL and Escape jail game interface

Using basic HTML code and JavaScript you can create a pretty good interface for playing grid games.The interface consists of a canvas for drawing the grid a paragraph for printing the messages in natural language as the game goes along, a title for the game and a set of buttons with different functions and a link towards another game. In the canvas we draw an 8x8 board. For every location that was visited we fill the grid cell with gray and for the current location we fill it with a color. For the cells that contain mines , messages , key or doors we draw an image using Image object in JavaScript. Using the arrow keys we change the current position in the grid using a 'keydown' event Listener.The user has the ability by using the mouse to flag cells that he thinks contains mines , but they are only for the user help this not being communicated with the server.

The logic of the game and the state : safe,unsafe,lost,won are decided by the python web server. Communication with the web server is realised using fetch api that requires the url of the resource , method to be used (GET,POST) and the cors mode in order to receive responses. The game uses only GET method because there isn't a lot of data it needs to send.

Using fetch api the game also loads the locations of props: messages ,mines , key and door.

When losing and winning an alert dialog box appears and the game is restarted

When the path visited is called by the interface the web server will call the method set_visited_fol that will tell the game master that a position was reached.

Whem the button Show Instructions is pressed a paragraph with instructions appear. when Show-Solution is pressed the board is redrawned and all the mines and messages are displayed.

When the button New board is pressed the game restart withj a new board by calling the function reset() from javascript gampe_page1.js sau game_page2.js

When the button Reset game is pressed the same board restarts

2.1.6 Python web servers

The web server was created in Python2.7 using the libraries BaseHttpServer with classes BaseHTTPRequestHandler ,HTTPServer and the library for regex re. For the logic of the game we also use the libraries os,re,sys,subprocess to make calls to prover9 and mace4 from command line.

In the method do_GET using regex library re we filter the path ,there are 8 paths for mineFOL and 10 for Escape game. For getting the props there are the paths: /messages,/bombs,/key and /door.The rest of 4 paths are used to communicate the state of the game with the server.

For communication about the state there are: /safe/i=(\ d+),j=(
d+) , /visited/i=(\d+),j=(\d+), /message/i=(\ d+),j=(\d+),/set_mine/i=(\d+),j=(\d+),/won,/lost.

The interface announces the web server at each step at what position it is, and asks the server if the position it wants to go to is safe or not, in the same time it communicates if it found a message or a bomb and asks if it won or lost.

There are two functional roles in the server : the player and the game master. The player will respond for the command safe and messsage,set_mine. And the game master for the commands visited,win and lost.

When the server is asked about the safety of a position it calls the method check_safe to check if mace4 can prove a location is safe and check_prop to check in the case it can't prove it is safe if there is a mine on that position. Both methods set their respective goals and call the method check_usingMace4 that will build the input file for the mace4 and will call mace4 to prove the theorems. If it succesed sends back a json object that on field safe it contains the string true and false otherwise.

When the server is asked if the game is won the server calls the function check_win and when when lost check_lost. These two methods also call the method check_usingMace4 but the assumptions used to check if the game is lost or won is in other container than the assumptions used to check if the position is safe or not , so there are two roles : the player and the game master, that have different informations about the world and different functions.

2.1.7 Prover9 and Mace4 library implementation

To call Prover9 and Mace4 we use the method call from the subprocess library.There are 3 methods that all call Prover9 and Mace4: check_usingMace4,check_problem and get_models. The method check_problem and check_usingMace4 are both used to check theorems but check_problem uses primarly prover9 and sometimes mace4 while check_usingMace4 uses only mace4. The method get_models uses mace4 to return models created by mace4.

In order to check theorems and build models you need to call Prover9 or Mace4 , to be able to call them the implementation builds input and output files.

the input file is created using the methods:build_file and build_file2. The build_file is used to build input file for Prover9 and build_file2 for Mace4, the main difference is that build—-file adds the goals sections while build_file2 does not.

A Mace4/Prover9 input file is composed from multiple parts or sections.The implemnetations models these sections as lists of strings. They are :options1,options2,assumptions and lists. The section modeled by options1 represent the options that use the keyword assign ,the one of options2 that use the keyword set.

There are methods to add and remove clauses for each section of an input file: add_rule,add_fact,add_goal,add_option1, add_option2, remove_goals,remove_assumptions,remove_options,remove_fact.

The assumptions have 2 methods that do essentially the same thing: add_rule,add_fact that add an assumption in the assumption section.The methods regarding this section give the possibility to add and remove assumptions in other lists or knowledge bases than the default knowledge base.

2.1.8 Running the games

The games are made from 2 parts: the front-end interface and the backend or the webServer

The interfaces are found in ther folder: MineFolInterface and the servers in the folder MineFolServer

In order to run MineFol game you need to first start the web server in the linux terminal using the command: python2.7 ./webServer.py than start the web Interface by opening in a browser page.html

To run Escape Jail game, you need to first start the web server in the linux terminal using the command: python2.7 ./webServerEscape.py than start the web Interface by opening in a browser page2.html

By default the games will search for a random game version from the folder: "games_configurations" and than will start the game using that version.This means it will load the mines, messages and other props from the files that end in with the name of the version.For example the version "_3" means that the file for mines will be: "bomb_3" ,for messages will be "messages_3" and for Escape Jail game it will be:"key_door_3"

For opening the servers the programs accepts options from the terminal in order to customize the game that is to be played by either providing with source files for mines, messages, key and door or by specifying a version to be played from the default folder: "games_configurations".

There are 3 options for the MineFol game and 4 for Escape Jail game. The options for MineFol game are: -b for mines source file, -m for messages source file and -v for specifying a specific version from a configuration existing in the configurations folder. The options for Escape Jail are: -b for mines source file, -m for messages source file , -k_d foe key and door source file , -v for specifying a specific version from a configuration existing in the configurations folder.

By pressing the button from the interface labeled "New Board" a new version will be chosen at random from the ones existing in the configurations folder.

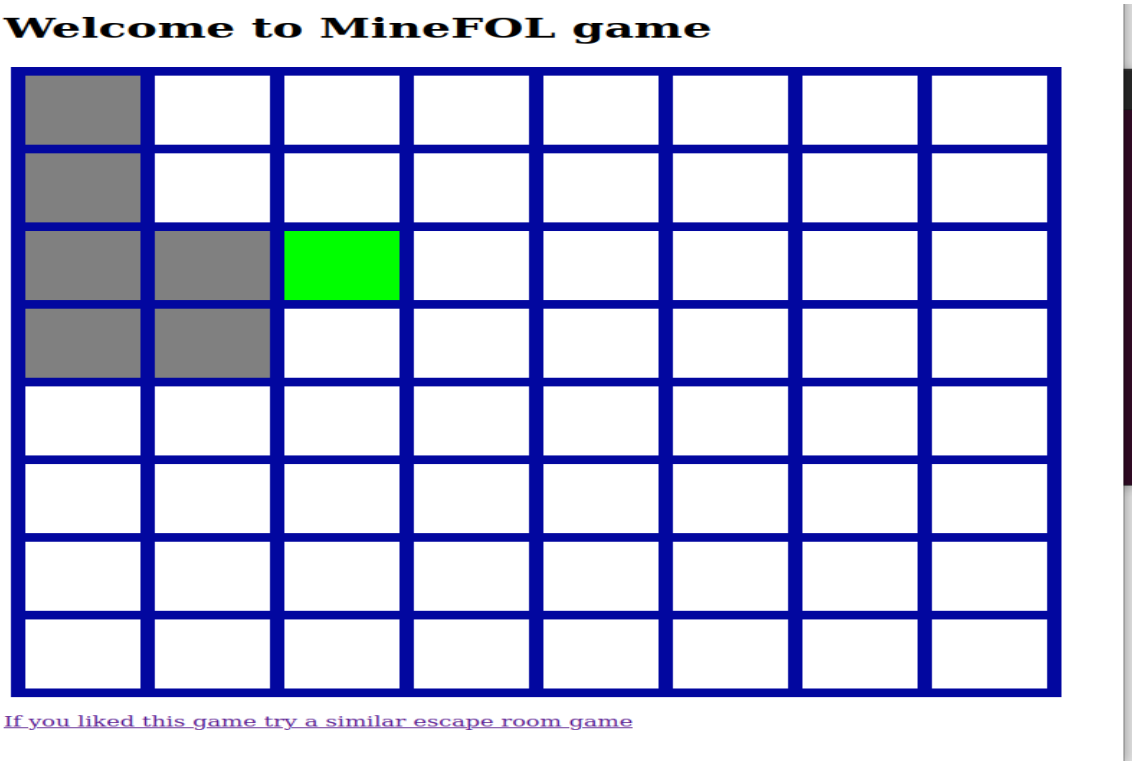
For the interface beside the classical method of opening their is the option to start a http server from the folder of the interfaces ,where someone must also chose a port different from the default:8080 and from 8085 because they are used by the web servers. A favorite option is using the port:4200.

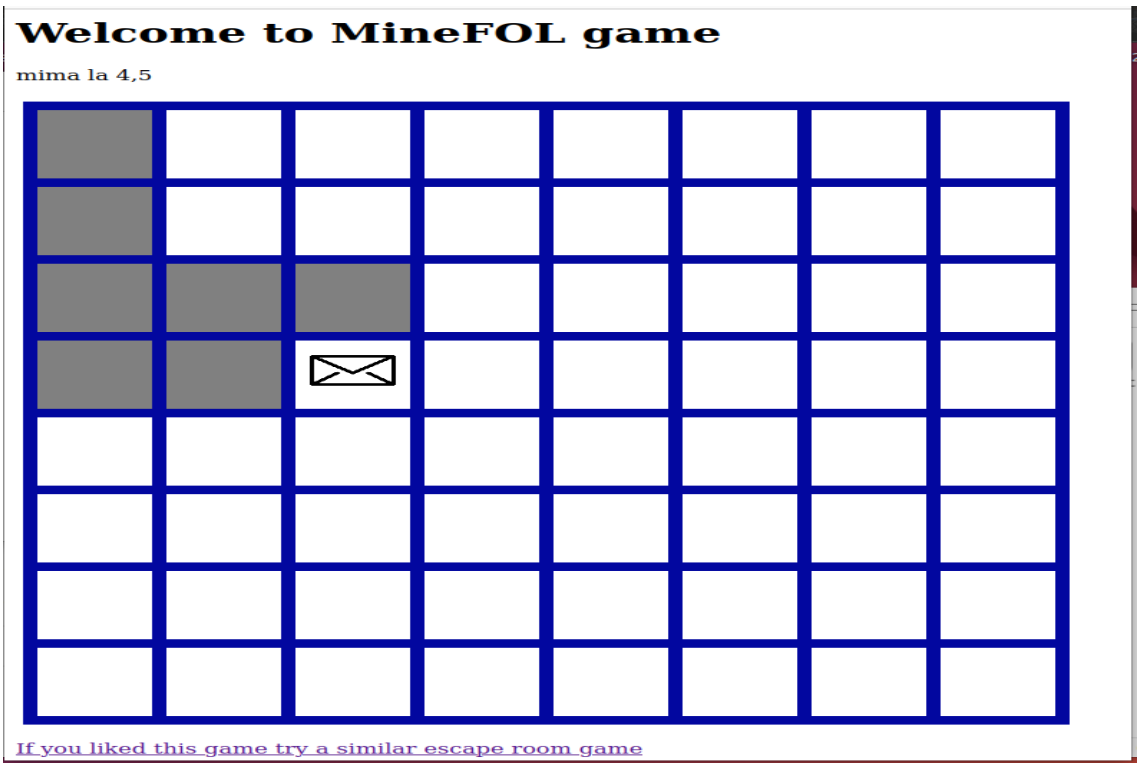
2.1.9 Other games options

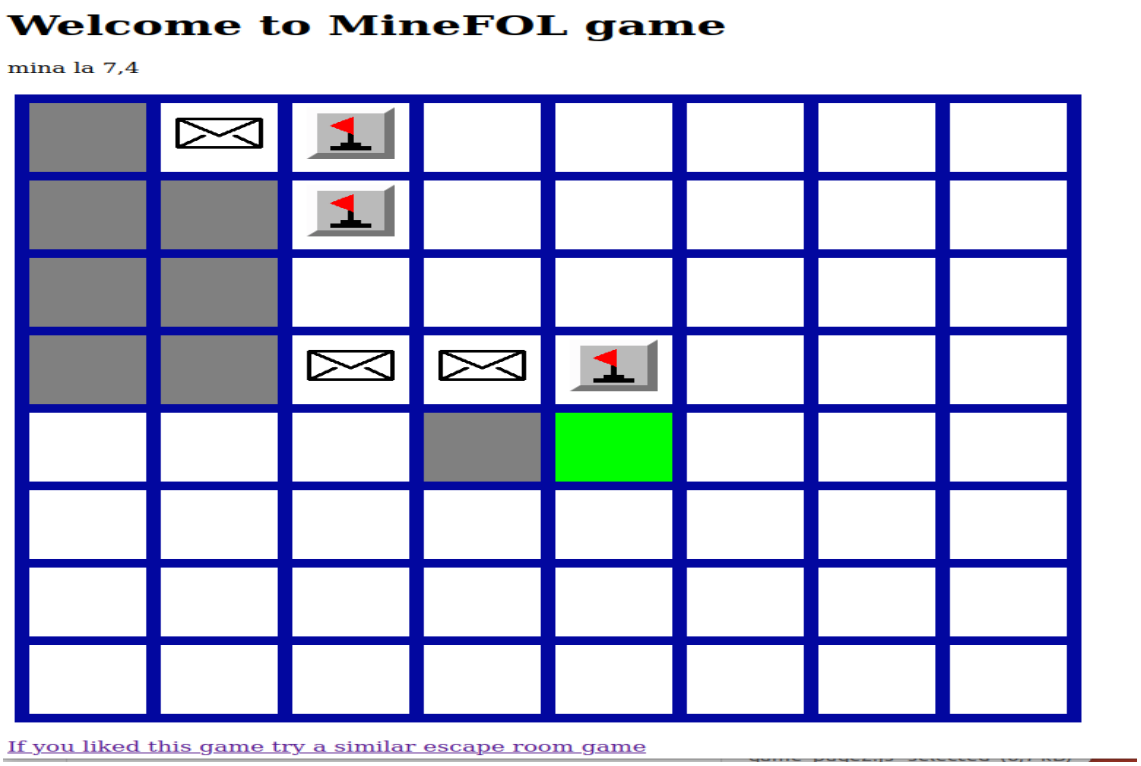
In the folder MineFolServer there is a module called "autogenerate_valid_game.py" which will generate a set of source files for a new game configurations. By default it will generate version "_3" but by specifying in the terminal other version name it will use that version, The generator creates always a configuration with 4 mines and 10 messages some that can also repeat.

2.1.10 Results

Interfaces




Welcome to MineFOL game
mima la 4,5
If you liked this game try a similar escape room game


Welcome to MineFOL game
mina la 7,4
If you liked this game try a similar escape room game


Welcome to MineFOL game
mina la 7,4
file://
You lost, please try again!
OK
If you liked this game try a similar escape room game

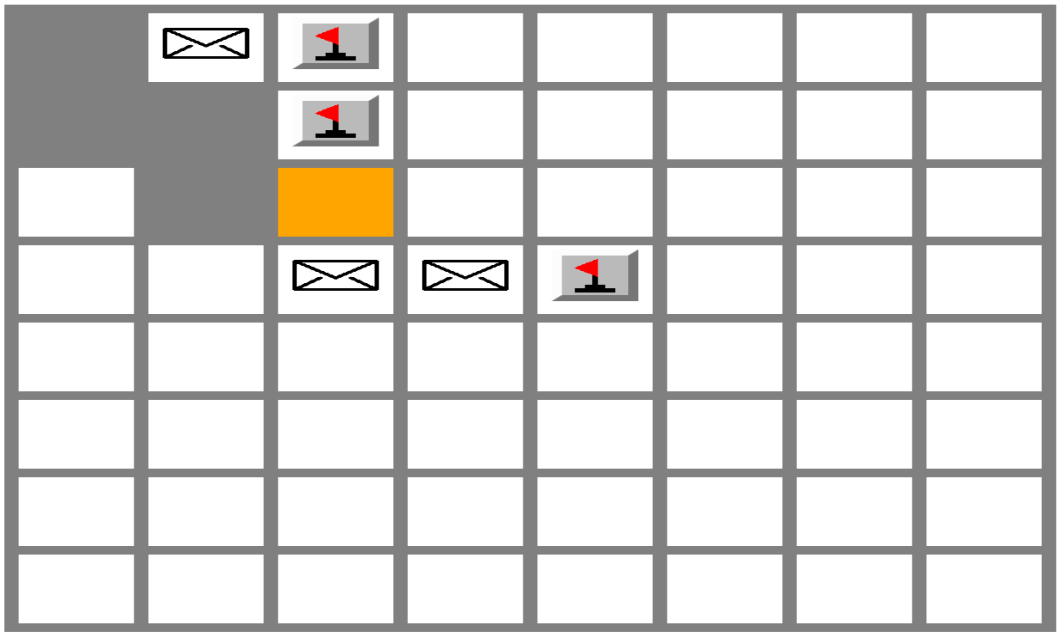

Welcome to MineFOL game
nicio mina pe diagonala
file://
You won , please play again!
OK
If you liked this game try a similar escape room game


Welcome to MineFOL game
mina pe coloana 3 daca y<4 si x!=3
file://
Prover9 detactaed a bomb to the position you want to go, so we will block your moving
OK
If you liked this game try a similar escape room game


Welcome to Escape from prison game
If you liked this game try a similar mineFOL game

Welcome to Escape from prison game

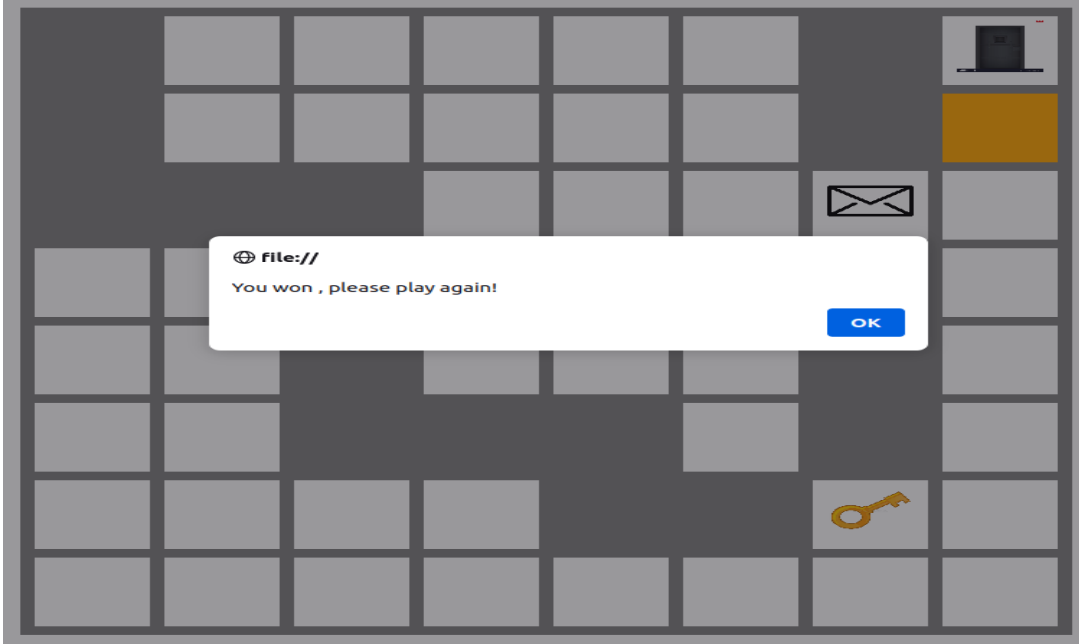
mina pe coloana 3 daca y<4 si x!=3



If you liked this game try a similar mineFOL game

Welcome to Escape from prison game

nicio mina pe diagonala



If you liked this game try a similar mineFOL game

Mace4 input file Minefol:
assign(domain_size,9).
set(arithmetic).
formulas(assumptions).

exists x exists y (x¿=1 & y¿=1 & key(x,y) & visited(x,y) & door(x,y))) ¡-¿ door_found.
door(x,y) & door(z,w) -¿ x=z & y=w.
key(x,y) & key(z,w) -¿ x=z & y=w.
exists x exists y (x¿=1 & y¿=1 & (visited(x,y) & mine(x,y)))-¿ lose.
door_found & key_found ¡-¿ Win.
key(7,7).
door(1,8).
-visited(1,1).
-visited(1,2).
-visited(1,5).
-visited(1,6).
-visited(1,7).
-visited(1,8).
-visited(2,3).
-visited(2,5).
-visited(2,6).
-visited(2,7).
-visited(2,8).
-visited(3,1).
-visited(3,5).
-visited(3,6).
-visited(3,7).
-visited(3,8).
-visited(4,1).
-visited(4,2).
-visited(4,3).
-visited(4,4).

-visited(4,5).
-visited(4,6).
-visited(4,7).
-visited(4,8).
-visited(5,1).
-visited(5,2).
-visited(5,3).
-visited(5,4).
-visited(5,5).
-visited(5,6).
-visited(5,7).
-visited(5,8).
-visited(6,1).
-visited(6,2).
-visited(6,3).
-visited(6,4).
-visited(6,5).
-visited(6,6).
-visited(6,7).
-visited(6,8).
-visited(7,1).
-visited(7,2).
-visited(7,3).
-visited(7,4).
-visited(7,5).
-visited(7,6).
-visited(7,7).
-visited(7,8).
-visited(8,1).
-visited(8,2).
-visited(8,3).
-visited(8,4).
-visited(8,5).
-visited(8,6).
-visited(8,7).
-visited(8,8).
-mine(1,1).
-mine(1,2).
mine(1,3).
-mine(1,4).
-mine(1,5).
-mine(1,6).
-mine(1,7).
-mine(1,8).
-mine(2,1).
-mine(2,2).
mine(2,3).
-mine(2,4).
-mine(2,5).
-mine(2,6).
-mine(2,7).
-mine(2,8).
-mine(3,1).
-mine(3,2).
-mine(3,3).
-mine(3,4).

-mine(3,5).
-mine(3,6).
-mine(3,7).
-mine(3,8).
-mine(4,1).
-mine(4,2).
-mine(4,3).
-mine(4,4).
mine(4,5).
-mine(4,6).
-mine(4,7).
-mine(4,8).
-mine(5,1).
-mine(5,2).
-mine(5,3).
-mine(5,4).
-mine(5,5).
-mine(5,6).
-mine(5,7).
-mine(5,8).
-mine(6,1).
-mine(6,2).
-mine(6,3).
-mine(6,4).
-mine(6,5).
-mine(6,6).
-mine(6,7).
-mine(6,8).
-mine(7,1).
-mine(7,2).
-mine(7,3).
mine(7,4).
-mine(7,5).
-mine(7,6).
-mine(7,7).
-mine(7,8).
-mine(8,1).
-mine(8,2).
-mine(8,3).
-mine(8,4).
-mine(8,5).
-mine(8,6).
-mine(8,7).
-mine(8,8).
visited(2,1).
visited(2,2).
visited(3,2).
visited(3,3).
visited(3,4).
visited(2,4).
visited(1,4).
visited(1,3).
-lose.
end_of_list.

Input file Escape jail gome master assign(domain_size,9).

set(arithmetic).
formulas(assum
ptions).
exists x exists y (x¿=1 & y¿=1 & key(x,y) & visited(x,y)) ¡-¿ key_found .
exists x (exists y (x¿=1 & y¿=1 & visited(x,y) & door(x,y))) ¡-¿ door_found.
door(x,y) & door(z,w) -¿ x=z & y=w.
key(x,y) & key(z,w) -¿ x=z & y=w.
exists x exists y (x¿=1 & y¿=1 & (visited(x,y) & mine(x,y)))-¿ lose.
door_found & key_found ¡-¿ Win.
key(7,7).
door(1,8).
-visited(1,1).
-visited(1,2).
-visited(1,3).
-visited(1,4).
-visited(1,5).
-visited(1,6).
-visited(1,7).
-visited(1,8).
-visited(2,2).
-visited(2,3).
-visited(2,4).
-visited(2,5).
-visited(2,6).
-visited(2,7).
-visited(2,8).
-visited(3,2).
-visited(3,3).
-visited(3,4).
-visited(3,5).
-visited(3,6).
-visited(3,7).
-visited(3,8).
-visited(4,4).
-visited(4,5).
-visited(4,6).
-visited(4,7).
-visited(4,8).
-visited(5,1).
-visited(5,2).
-visited(5,6).
-visited(5,7).
-visited(5,8).
-visited(6,1).
-visited(6,2).
-visited(6,3).
-visited(6,4).
-visited(6,6).
-visited(6,7).
-visited(6,8).
-visited(7,1).
-visited(7,2).
-visited(7,3).
-visited(7,6).
-visited(7,7).
-visited(7,8).

-visited(8,1).
-visited(8,2).
-visited(8,3).
-visited(8,6).
-visited(8,7).
-visited(8,8).
-mine(1,1).
-mine(1,2).
mine(1,3).
-mine(1,4).
-mine(1,5).
-mine(1,6).
-mine(1,7).
-mine(1,8).
-mine(2,1).
-mine(2,2).
mine(2,3).
-mine(2,4).
-mine(2,5).
-mine(2,6).
-mine(2,7).
-mine(2,8).
-mine(3,1).
-mine(3,2).
-mine(3,3).
-mine(3,4).
-mine(3,5).
-mine(3,6).
-mine(3,7).
-mine(3,8).
-mine(4,1).
-mine(4,2).
-mine(4,3).
-mine(4,4).
mine(4,5).
-mine(4,6).
-mine(4,7).
-mine(4,8).
-mine(5,1).
-mine(5,2).
-mine(5,3).
-mine(5,4).
-mine(5,5).
-mine(5,6).
-mine(5,7).
-mine(5,8).
-mine(6,1).
-mine(6,2).
-mine(6,3).
-mine(6,4).
-mine(6,5).
-mine(6,6).
-mine(6,7).
-mine(6,8).
-mine(7,1).
-mine(7,2).

-mine(7,3).
mine(7,4).
-mine(7,5).
-mine(7,6).
-mine(7,7).
-mine(7,8).
-mine(8,1).
-mine(8,2).
-mine(8,3).
-mine(8,4).
-mine(8,5).
-mine(8,6).
-mine(8,7).
-mine(8,8).
visited(2,1).
visited(3,1).
visited(4,1).
visited(4,2).
visited(4,3).
visited(5,3).
visited(5,4).
visited(5,5).
visited(6,5).
visited(7,5).
visited(8,5).
visited(8,4).
visited(7,4).
-lose.
end_of_list.

Input file player MineFOL
assign(domain_size,9).
set(arithmetic).
formulas(assumptions).
mine(x,y) — safe(x,y).
mine(x,y)-¿-safe(x,y).
safe(1,1).
(y¡4 & y!=3) -¿ mine(y,3).
mine(4,5).
mine(7,4).
safe(4,3).
end_of_list.

Output file mace4 ========== Mace4 ==========
Mace4 (64) version 2017-11A (CIIRC), November 2017. Process 13445 was started by alex on alex-IdeaPad-Gaming-3-15ACH6, Wed Dec 7 00:35:40 2022 The command was "mace4 -m 3 -n 9 -c -f mace4_builder.in". ========== end of head ==========
========== INPUT ==========
assign(domain_size,9). set(arithmetic).
formulas(assumptions). mine(x,y) — safe(x,y). mine(x,y) -¿ -safe(x,y). safe(1,1). y ¡ 4 & y != 3 -¿ mine(y,3). mine(4,5). mine(7,4). safe(4,3). end_of_list.
========== end of input ==========
========== PROCESS NON-CLAUSAL FORMULAS ==========
1 mine(x,y) -¿ -safe(x,y) # label(non_clause). [assumption]. 2 y ¡ 4 & y != 3 -¿ mine(y,3) # label(non_clause). [assumption].
========== end of process non-clausal formulas ===

========== CLAUSES FOR SEARCH ==========

formulas(mace4_clauses). mine(x,y) — safe(x,y). -mine(x,y) — -safe(x,y). safe(1,1). -(x ¡ 4) — 3 = x — mine(x,3). mine(4,5). mine(7,4). safe(4,3). end_of_list.

========== end of clauses for search ==========

========== DOMAIN SIZE 9 ==========

========== MODEL ==========

interpretation(9, [number=1, seconds=0], [

relation(mine(_,_), [
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0]),

relation(safe(_,_), [
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1])
]).

========== end of model ==========

========== MODEL ==========

interpretation(9, [number=2, seconds=0], [

relation(mine(_,_), [
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1]),

relation(safe(_,_), [
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 0])
]).

========== end of model ==========

========== MODEL ==========

interpretation(9, [number=3, seconds=0], [

relation(mine(_,_), [
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0]),

relation(safe(_,_), [
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 1])
]).

========== end of model ==========

========== STATISTICS ==========

For domain size 9.

Current CPU time: 0.00 seconds (total CPU time: 0.00 seconds).
Ground clauses: seen=175, kept=169.
Selections=75, assignments=77, propagations=91, current_models=3.
Rewrite_terms=0, rewrite_bools=259, indexes=0.
Rules_from_neg_clauses=0, cross_offs=0.

========== end of statistics ==========

User_CPU=0.00, System_CPU=0.00, Wall_clock=0.

Exiting with 3 models.

Process 13445 exit (max_models) Wed Dec 7 00:35:40 2022
The process finished Wed Dec 7 00:35:40 2022

3 A3:Planning

4 Appendices

4.1 A1 Appendix

4.1.1 Original code

```
def weightedAstarSearch(problem, heuristic=nullHeuristic, weightE=2, weightC=1):
    """Search the node that has the lowest combined cost and heuristic first."""
    "*** YOUR CODE HERE ***"

    print("Wastar")
    frontier = PriorityQueue()
    extended = []
    actions = Queue()
    frontier.push((problem.getStartState(), 0, 0, 0), 0)
    while not frontier.isEmpty():
        node = frontier.pop()
        extended.append(node)
        print("number of corners not riched",len(problem.remainingCorners))
        if problem.isGoalState(node[0]):
            print("goal riched")
            return build_path(extended,problem.visitedCorners)
        vecini = problem.getSuccessors(node[0])
        print("vecini",vecini)
        for v in vecini:
               bool = False
               for e in extended:
                  if e[0] == v[0]:
                       bool = True
               if bool == False:
                print("can add mmore")
                costTotal = node[3] + v[2]
                heuristicCost = weightE * heuristic(v[0], problem) + weightC * costTotal
                frontier.push((v[0], v[1], node[0], costTotal), heuristicCost)
    print("no new nodes")
    return []
def DinamicWeightedAstarSearch(problem, heuristic=nullHeuristic):
    """Search the node that has the lowest combined cost and heuristic first."""
    "*** YOUR CODE HERE ***"
    print("remainingCorners",problem.remainingCorners)
    print("DWastar")
    frontier = PriorityQueue()
    extended = []
    frontier.push((problem.getStartState(), 0, 0, 0), 0)
    depth=0;
    random.seed(datetime.now())
    epsilon=random.random();
   # len(breadthFirstSearch(problem))
    N=60
    extended = []
    problem.remainingCorners=[problem.corners[0],problem.corners[1],problem.corners[2],problem.corner
    pause();
    while not frontier.isEmpty():
```

```
        depth=depth+1;
        weightE=1+epsilon-epsilon*depth/N;
        print("weightE",weightE);
        node = frontier.pop()
        extended.append(node)
        if problem.isGoalState(node[0]):
          print("got to goal")
          print("extended",extended)
          pause();
          return build_path(extended,problem.visitedCorners,problem.getStartState())

        vecini = problem.getSuccessors(node[0])
        for v in vecini:
            bool = False
            for e in extended:
                if e[0] == v[0]:
                    bool = True
            if bool == False:
                costTotal = node[3] + v[2]
                heuristicCost = weightE * heuristic(v[0], problem) + costTotal
                frontier.push((v[0], v[1], node[0], costTotal), heuristicCost)



def build_path(extended,visitedCorners,start_point):
    paths = []
    actions = Queue()
    parent=(0,0,0,0)
    for vc in visitedCorners:
        path = Stack()

        for n in extended:
         if(n[0]==vc):
           print("n0",n)
           path.push(n)
           parent = n[0]
           break


        print("extended",extended)

        while not parent == 0:
            print("parent",parent)
            for n in extended:
                if n[0] == parent:
                    path.push(n)
                    parent = n[2]
                    break

        paths.append(path.list[1:])
    print("visited corners in order are:", visitedCorners)
    print("paths are:")
    for path1 in paths:
        print("path:", path1)
    noduri_legatura = [0,0,0,0,0,0,0]
    search_intersections=[1,2,3,4,5,6]
```

```
for i in paths[0]:
    for j in paths[1]:
        for k in paths[2]:
            for l in paths[3]:
                for m in range(len(search_intersections)):
                    if i[0] == j[0] and search_intersections[m]==1:
                        search_intersections[m]=0
                        noduri_legatura[1]=i

                    if i[0] == k[0] and search_intersections[m] == 4:
                        search_intersections[m] = 0
                        noduri_legatura[4]=j
                    if i[0] == l[0] and search_intersections[m] == 5:
                        search_intersections[m] = 0
                        noduri_legatura[5]=l
                    if j[0] == k[0] and search_intersections[m] == 2:
                        search_intersections[m] = 0
                        noduri_legatura[2]=k
                    if j[0] == l[0] and search_intersections[m] == 6:
                        search_intersections[m] = 0
                        noduri_legatura[6]=j
                    if l[0] == k[0] and search_intersections[m] == 3:
                        search_intersections[m] = 0
                        noduri_legatura[3]=l
print("noduri de legatura in arbore sunt:", noduri_legatura)
destination = start_point
for i in range(0,4):
 print("i=",i)
 current_possition = paths[i][0]

 #reverse a path:

 help_stack =Stack()
 help_stack2=Queue()
 print("starting point", current_possition )
 for n in paths[i]:
      print("destination to get to",destination)
      if n[0]==destination:

          print("destination",destination)
          help_stack=Stack()
          nod_to_go_back_to=0
          current_possiton=paths[i][0]
          if i==3:
            nod_to_go_back_to = noduri_legatura[i -1][0]
          else:
            nod_to_go_back_to = noduri_legatura[i +1 ][0]
          print("go  back to",nod_to_go_back_to)
          print("current", current_possition )
          destination= nod_to_go_back_to
     #     #you reached a point from where you will jump to an intersection so go back to that po
          while  not  (current_possition[0]==nod_to_go_back_to )and  not current_possition[2]==0:


            print("backtracking",current_possition)
            #reverse the action made to get to a corner
```

```
                if (current_possition[1] == 'North'):
                    help_stack.push("South")
                if (current_possition[1]  == 'South'):
                    help_stack.push("North")
                if (current_possition[1] == 'West'):
                    help_stack.push("East")
                if (current_possition[1] == 'East'):
                    help_stack.push('West')
                #go back to your parent
                for j in extended:
                    if j[0]==current_possition[2]:
                        current_possition=j
                        break
            print(" finished backtracking")


            break
        else:

            if n[1]!=0:
              print("n not path",n)
              help_stack2.push(n[1])


    actions.list=actions.list+help_stack2.list+help_stack.list
    print("actions after a path",actions.list)

   print("actions",actions.list)
   return actions.list
```

```
def CostOrientedWeightAStar(problem, heuristic=nullHeuristic, weightE=2, weightC=1):
    return weightedAstarSearch(problem, heuristic, 1, 2)


def HeuristicOrientedWeightAStar(problem, heuristic=nullHeuristic, weightE=2, weightC=1):
    return weightedAstarSearch(problem, heuristic, 2, 1)
    return weightedAstarSearch(problem, heuristic, 2, 1)

def greedyWeightAStar(problem, heuristic=nullHeuristic, weightE=2, weightC=1):
    return weightedAstarSearch(problem, heuristic ,1,0)
```

```
 def isGoalState(self, state):
        x,y =state
        for i  in range(len(self.remainingCorners)):
           x1,y1=self.remainingCorners[i]
           if(x==x1 and y==y1):
                self.visitedCorners.append(self.remainingCorners[i])
                self.remainingCorners.remove(self.remainingCorners[i])
                i=i-1
                break
        return len(self.remainingCorners) == 0
```

```
def cornersHeuristic1(state, problem):
    corners = problem.remainingCorners
    return len(corners)

def cornersHeuristic2(state, problem):
    # These are the corner coordinates
    x,y= state
    corners=problem.remainingCorners
    furthestcornerIndex=0
    biggestDistance= -1
    for index in xrange(0,len(corners)):
       xg,yg= corners[index]
       dist=abs(x-xg)+abs(x-xg)
       if dist>biggestDistance:
           biggestDistance =dist
           furthestcornerIndex=index
    xg, yg = corners[furthestcornerIndex]

    return abs(x-xg)*abs(y-yg)
```

```
def cornersHeuristic3(state, problem):
    corners = problem.remainingCorners
    # These are the corner coordinates
    x,y= state
    bigestDistanceBetweenCorners=0;
    closestCorner=999999;
    print("remaining corners",len(corners))
    for i in range(len(corners)):
         xc, yc = corners[i]
         dist_corners=mazeDistance((x,y),(xc,yc),problem.startingGameState)

         if(dist_corners<closestCorner):
             closestCorner=dist_corners

         for j in (i+1,len(corners)-1):
            if j<len(corners):
              xc2,yc2=corners[j]
              dist=mazeDistance((xc,yc),(xc2,yc2),problem.startingGameState)
```

```
            if dist >  bigestDistanceBetweenCorners:
               bigestDistanceBetweenCorners=dist

    return closestCorner+bigestDistanceBetweenCorners


def cornersHeuristic4(state, problem):
    corners = problem.remainingCorners
    # These are the corner coordinates
    x,y= state
    averageDistanceBetweenCorners=0;
    closestCorner=999999;

    for i in range(len(corners)):
         xc, yc = corners[i]
         dist_corners=mazeDistance((x,y),(xc,yc),problem.startingGameState)
         if(dist_corners<closestCorner):
             closestCorner=dist_corners

         for j in(i+1,len(corners)):
          if j < len(corners):
            xc2,yc2=corners[j]
            dist=mazeDistance((x,y),(xc,yc),problem.startingGameState)
            averageDistanceBetweenCorners +=dist

    return closestCorner+averageDistanceBetweenCorners/len(corners)



def cornersHeuristic5(state, problem):
    corners = problem.remainingCorners
    # These are the corner coordinates
    x,y= state
    averageDistanceBetweenCorners=0;
    closestCorner=999999;

    for i in range(len(corners)):
         xc, yc = corners[i]
         #=abs(x-xc)+abs(y-yc)
         dist_corners=abs(x-xc)+abs(y-yc)
         if(dist_corners<closestCorner):
             closestCorner=dist_corners

         for j in(i+1,len(corners)):
          if j < len(corners):
            xc2,yc2=corners[j]
            dist=abs(xc-xc2)+abs(yc-yc2)
            averageDistanceBetweenCorners +=dist

    return closestCorner+averageDistanceBetweenCorners/len(corners)
```

```
class AStarCornersAgent(SearchAgent):

    def __init__(self):
        self.searchFunction = lambda prob: search.aStarSearch(prob, cornersHeuristic1)
        self.searchType = CornersProblem

class AStarCornersAgentWeightedCO(SearchAgent):

    def __init__(self):
        self.searchFunction = lambda prob: search.CostOrientedWeightAStar(prob,cornersHeuristic1)
        self.searchType = CornersProblem

class AStarCornersAgentWeightedHO(SearchAgent):
     def __init__(self):
        self.searchFunction = lambda prob: search.HeuristicOrientedWeightAStar(prob,cornersHeuristic1
        self.searchType = CornersProblem


class GreedyCornersAgentHeuristic(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.greedyWeightAStar(prob, cornersHeuristic1)
        self.searchType = CornersProblem


class DinamicCornersAgentHeuristic(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.DinamicWeightedAstarSearch(prob, cornersHeuristic1)
        self.searchType = CornersProblem







class AStarCornersAgenth2(SearchAgent):

    def __init__(self):
        self.searchFunction = lambda prob: search.aStarSearch(prob, cornersHeuristic2)
        self.searchType = CornersProblem

class AStarCornersAgentWeightedCO2(SearchAgent):

    def __init__(self):
        self.searchFunction = lambda prob: search.CostOrientedWeightAStar(prob,cornersHeuristic2)
        self.searchType = CornersProblem


class AStarCornersAgentWeightedHO2(SearchAgent):
     def __init__(self):
        self.searchFunction = lambda prob: search.HeuristicOrientedWeightAStar(prob,cornersHeuristic2
        self.searchType = CornersProblem


class GreedyCornersAgentHeuristic2(SearchAgent):
```

```
    def __init__(self):
        self.searchFunction = lambda prob: search.greedyWeightAStar(prob, cornersHeuristic2)
        self.searchType = CornersProblem

class DinamicCornersAgentHeuristic2(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.DinamicWeightedAstarSearch(prob, cornersHeuristic2)
        self.searchType = CornersProblem


class AStarCornersAgenth3(SearchAgent):

    def __init__(self):
        self.searchFunction = lambda prob: search.aStarSearch(prob, cornersHeuristic3)
        self.searchType = CornersProblem

class AStarCornersAgentWeightedCO3(SearchAgent):

    def __init__(self):
        self.searchFunction = lambda prob: search.CostOrientedWeightAStar(prob,cornersHeuristic3)
        self.searchType = CornersProblem


class AStarCornersAgentWeightedHO3(SearchAgent):
     def __init__(self):
        self.searchFunction = lambda prob: search.HeuristicOrientedWeightAStar(prob,cornersHeuristic3
        self.searchType = CornersProblem


class GreedyCornersAgentHeuristic3(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.greedyWeightAStar(prob, cornersHeuristic3)
        self.searchType = CornersProblem

class DinamicCornersAgentHeuristic3(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.DinamicWeightedAstarSearch(prob, cornersHeuristic3)
        self.searchType = CornersProblem


class AStarCornersAgenth4(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.aStarSearch(prob, cornersHeuristic4)
        self.searchType = CornersProblem


class AStarCornersAgentWeightedCO4(SearchAgent):

    def __init__(self):
        self.searchFunction = lambda prob: search.CostOrientedWeightAStar(prob,cornersHeuristic4)
        self.searchType = CornersProblem


class AStarCornersAgentWeightedHO4(SearchAgent):
     def __init__(self):
        self.searchFunction = lambda prob: search.HeuristicOrientedWeightAStar(prob,cornersHeuristic4
```

```
        self.searchType = CornersProblem


class GreedyCornersAgentHeuristic4(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.greedyWeightAStar(prob, cornersHeuristic4)
        self.searchType = CornersProblem

class DinamicCornersAgentHeuristic4(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.DinamicWeightedAstarSearch(prob, cornersHeuristic4)
        self.searchType = CornersProblem

class AStarCornersAgenth5(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.aStarSearch(prob, cornersHeuristic5)
        self.searchType = CornersProblem


class AStarCornersAgentWeightedCO5(SearchAgent):

    def __init__(self):
        self.searchFunction = lambda prob: search.CostOrientedWeightAStar(prob, cornersHeuristic5)
        self.searchType = CornersProblem


class AStarCornersAgentWeightedHO5(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.HeuristicOrientedWeightAStar(prob, cornersHeuristic
        self.searchType = CornersProblem


class GreedyCornersAgentHeuristic5(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.greedyWeightAStar(prob, cornersHeuristic5)
        self.searchType = CornersProblem


class DinamicCornersAgentHeuristic5(SearchAgent):
    def __init__(self):
        self.searchFunction = lambda prob: search.DinamicWeightedAstarSearch(prob, cornersHeuristic5)
        self.searchType = CornersProblem
```

4.2 A2 Appendix

4.2.1 Original code

```
from __future__  import print_function
import re
import subprocess

options1 =[]
options2 =[]
# this is the default base for assumptions another one can be used
assumptions =[]
goals =[]
lists =[]

builder= lambda  :  print ("no knowledge function given")


def build_file(path ,diferent_base=None):
    f = open(path, "w")

    for option in options1:
        f.write("assign(" + option + ").\n")

    for option in options2:
        f.write("set(" + option + ").\n")

    #
    # f.write("list(distinct).\n")
    # for list in lists:
    #     f.write(list+"." + "\n")
    # f.write("end_of_list.\n")

    f.write("formulas(assumptions).\n")

    if diferent_base==None:
      for assumption in assumptions:
        f.write(assumption + "\n")
    else:
      for assumption in diferent_base:
         f.write(assumption + "\n")



    f.write("end_of_list.\n")

    f.write("formulas(goals).\n")
    for goal in goals:  # add
      f.write(goal + "\n")
    f.write("end_of_list.\n")
    # add goals   f.close()
    f.close()

def build_file2(path ,diferent_base=None):
    f = open(path, "w")
```

```
    for option in options1:
        f.write("assign(" + option + ").\n")

    for option in options2:
        f.write("set(" + option + ").\n")

    #
    # f.write("list(distinct).\n")
    # for list in lists:
    #     f.write(list+"." + "\n")
    # f.write("end_of_list.\n")

    f.write("formulas(assumptions).\n")
    if diferent_base==None:
      for assumption in assumptions:
        f.write(assumption + "\n")
    else:
        for assumption in diferent_base:
            f.write(assumption + "\n")
    for goal in goals:
        f.write(goal + "\n")
    f.write("end_of_list.\n")
    # add goals
    f.close()

def load_assumptions(filePath ,diferent_base=None):
    f=open (filePath)
    lines=f.readlines()
    for line in lines:
        add_fact(line,diferent_base)

def set_goal(goal):
    goals[:]=[]
    add_goal(goal)

def set_goals(goals2):
    goals[:]=[]
    for goal in goals2:
        add_goal(goal)



def add_rule(assumption,diferent_base=None):
    assumption=assumption+"."
    if diferent_base==None:
        assumptions.append(assumption)
    else:
        diferent_base.append(assumption)

def add_fact(assumption,diferent_base=None):
    assumption=assumption+"."
    if diferent_base == None:
        assumptions.append(assumption)
    else:
        diferent_base.append(assumption)
```

```
def add_goal(goal):
    goal=goal+"."
    goals.append(goal)

def add_option1(option):
    options1.append(option)
def add_option2(option):
    options2.append(option)



def remove_goals():
    goals[:]=[]

def remove_assumptions(diferent_base=None):
    if diferent_base==None:
        assumptions[:]=[]
    else:
        diferent_base[:]=[]



def remove_options():
    options1[:]=[]
    options2[:]=[]


def remove_fact(assumption,different_base=None):
 if different_base==None:
   if assumption+"." in assumptions:
     assumptions.remove(assumption+".")
 else:
   if assumption+"." in different_base:
     different_base.remove(assumption+".")




#used for inference in this project
def check_usingMace4(name="mineFOL",different_base=None,domain=9,build=None):
    print("Name of problem is", name)
    path = "mace4_builder.in"
    patho = "mace4_model.out"
    if not different_base==None:
     path = "mace4_builder_master.in"
     patho = "mace4_model_master.out"

    list_cmd = []
    models = []
    nr_models=0
    if not build == None:
        remove_options()
        remove_assumptions(different_base)
        remove_goals()
        build()
```

```
program="mace4"
list_cmd.append(program)
list_cmd.append("-m")
list_cmd.append("3")
list_cmd.append("-n")
list_cmd.append(str(domain))
list_cmd.append("-c")
list_cmd.append("-f")
list_cmd.append(path)

g = open(patho, "w")
build_file2(path,different_base)
subprocess.call(list_cmd, stdout=g)
g.close()
g = open(patho, "r")
res = g.read()
found = False
if program=="mace4":
  result = re.search(r"\nExiting with (\d+) models.\n", res)
  if result:
    print("somehow result")
    nr_models=int(result.group(1))
    string_start="============================== MODEL ================================="
    string_end="============================== end of model ============================"
    string_antet="interpretation\\((.*?)\\], \\["
    string_model="([^\s].+)+"
    result2=re.findall(string_model,res)

    if result2:
      model1 = []
      start=False
      for r in result2:
        # print(r)
        rs=re.search(string_start,r)
        if rs:
          start=True
        else:
          if start==True:
              rE = re.search(string_end, r)
              if rE:
                  start = False
                  print("mod el", model1)
                  models.append(model1)
                  model1 = []
              else:
                 ra=re.search(string_antet,r)
                 if not ra:
                   string_begin_line="relation\\((.*?), \\[(.*?)\ \]"
                   result3 = re.search(string_begin_line, r)
                   if not result3==None:
                      model1.append((result3.group(1),int(result3.group(2))))
  # print ("model found:", models)
  if nr_models>=1:
      return True
  else:
      return False
```

```
#getting all models mace4 can produce not to be used for inference ,switches on prover9 if given goal
def get_models(name,different_base=None,domain=2 , build=None):
    print("Name of problem is", name)
    path = "mace4_builder.in"
    patho = "mace4_model.out"
    if not different_base==None:
     path = "mace4_builder_master.in"
     patho = "mace4_model_master.out"

    list_cmd = []
    models=[]
    nr_models=0
    if not build == None:
        remove_options()
        remove_assumptions(different_base )
        remove_goals()
        build()
    program="mace4"
    if len(goals) > 0:
        print("Gaols were given routed on prover9")
        program="prover9"
        list_cmd.append(program)

    else:
        list_cmd.append(program)
        list_cmd.append("-m")
        list_cmd.append("-1")
        list_cmd.append("-n")
        list_cmd.append(str(domain))
        list_cmd.append("-c")

    g = open(patho, "w")
    build_file(path,different_base)
    list_cmd.append("-f")
    list_cmd.append(path)
    subprocess.call(list_cmd, stdout=g)
    g.close()
    g = open(patho, "r")
    res = g.read()
    found = False

    if program=="mace4":
      print("model found:", models)
      result = re.search( '"r " \nExiting with (\d+) models.\n""', res)
      if result:
        print("groups are", result.groups())
        print("group i searched for is:", result.group(1))
        nr_models=int(result.group(1))
        string_start="============================== MODEL ================================="
        string_end="============================== end of model ==========================="
        string_antet="interpretation\\((.*?)\\], \\["
        string_model="([^\s].+)+"
        result2=re.findall(string_model,res)
```

```
        if result2:
          model1 = []
          start=False
          for r in result2:
            print(r)
            rs=re.search(string_start,r)
            if rs:
              start=True
            else:
              if start==True:
                  rE = re.search(string_end, r)
                  if rE:
                      start = False
                      print("model", model1)
                      models.append(model1)
                      model1 = []
                  else:
                     ra=re.search(string_antet,r)
                     if not ra:
                        string_begin_line="relation\\((.*?), \\[(.*?)\\]"
                        result3 = re.search(string_begin_line, r)
                        if result3:
                          model1.append((result3.group(1),int(result3.group(2))))
                        else:
                           string_begin_line="function\\((.*?), \\[(.*?)\\]"
                           result3 = re.search(string_begin_line, r)
                           if result3:
                            model1.append((result3.group(1),int(result3.group(2))))

      print("model found:", models)

    else:
        found = res.find("\nTHEOREM PROVED\n")
        if found > 0:
            found = True
        else:
            found = False

        return found

    return (nr_models,models)

#used to check using prover9 ,not used at this project
def check_problem(name,different_base=None,build=None):

    print("Name of problem is",name)
    path="prover_provers.in"
    patho="prover_proved.out"
    list_cmd=[]

    if not build==None:
        remove_options()
        remove_assumptions(different_base)
        remove_goals()
        build()
```

```
    program = "prover9"
    if len(goals)==0:
     program="mace4"
     list_cmd.append(program)
     list_cmd.append("-m")
     list_cmd.append("-1")
     list_cmd.append("-n")
     list_cmd.append("2")
     list_cmd.append("-c")
    else:
      list_cmd.append(program)
    g=open(patho,"w")

    build_file(path,different_base)
    list_cmd.append("-f")
    list_cmd.append(path)
    subprocess.call(list_cmd,stdout=g)
    g.close()
    g = open(patho, "r")
    res=g.read()
    found=False
    if program == "prover9":
      found=res.find("\nTHEOREM PROVED\n")
      if found>0:
        found=True
      else:
        found=False
    else:

        result=re.search(r"\nExiting with (\d+) models.\n",res)

        if result:
          print("groups are",result.groups())
          print("group i searched for is:",result.group(1))
          found=True
        else:
          found=False
    print(found)
    return found




def exists(domain,rule):
  for dom in domain :
      if rule(dom):
          return True
  return False

def all(domain,rule):
    for dom in domain:
         if not rule(dom):
             return False
    return True
```

```
def isTautology(rule,nr_variables ,domain_of_variables=[0,1 ], variables=[]):
       var=False
       for d in domain_of_variables:
            variables.append(d)
            if nr_variables==1:
               if not rule(variables):
                   return  False
               else:
                   return True
            else:
                var=var and isTautology(rule, nr_variables-1,domain_of_variables,variables)
       return var



def isPureLiteral(rules,nr_variables_rules,nr_rules,nr_literal_rules,domain_variables=[ 0,1]):
    return False



#propety class
class Propety:
   def __init__(self,dim,name):
    self.myName=name
    self.locations=[]
    self.dim=dim
    for i in range(1,dim+1):
        rowList=[]
        for j in range(1,dim+1):
          rowList.append([0,""])
        self.locations.append(rowList)

   def at(self,i,j):
      return self.locations[i-1][j-1]
   def set(self,i,j,msg):
        self.locations[i-1][j-1][0]=1
        self.locations[i-1][j-1][1] = msg

   def set_state(self, i, j,nr=-1):
       self.locations[i - 1][j - 1][0] =nr
       self.locations[i - 1][j - 1][1] = ""




   def take_prop(self,i,j):
    if not self.locations[i-1][j-1][0]==0:
        return self.locations[i-1][j-1][1]
    else:
        return ""

#Common code for both MineFol and Escape Game
from prover9_lib import add_option1, remove_fact
from prover9_lib import add_option2
```

```
from prover9_lib import add_fact
from prover9_lib import add_rule
from prover9_lib import check_problem
from prover9_lib import get_models
from prover9_lib import add_goal
from prover9_lib import builder
from prover9_lib import set_goal
from prover9_lib import set_goals
from prover9_lib import load_assumptions
from prover9_lib import check_usingMace4
from propety import Propety
from prover9_lib import remove_assumptions
from prover9_lib import remove_goals
from prover9_lib import remove_options
from util import *
from time import *
import re



def load_messages(filePath,prop,prop2):
    f=open(filePath)
    lines=f.readlines()
    form="(\d+) (\d+) (.+)\.(.+)"
    for line in lines:
     result= re.search(form,line)
     print("result:",result.group(1)," ; ",result.group(2)," ; ",result.group(3),";",result.group(4))
     prop2.set(int(result.group(1)),int(result.group(2)),result.group(4))
     prop.set(int(result.group(1)),int(result.group(2)),result.group(3))
    print(prop.myName + " loaded")

def load_obstacles(filePath,obst):
    f=open(filePath)
    lines=f.readlines()
    form="(\d+) (\d+)"
    for line in lines:
     result= re.search(form,line)
     obst.set(int(result.group(1)),int(result.group(2)),"")
    print(obst.myName + " loaded")

def check_safe(x,y,name="mine_fol"):
    set_goal("safe("+str(x)+","+str(y)+")")
    return check_usingMace4(name)

def check_prop(x,y,different_base=None,prop_name="mine",name="mine_fol"):
    set_goal(prop_name+"(" + str(x) + "," + str(y) + ")")
    return check_usingMace4(name,different_base=None)

def set_obst(i,j,obst_name="mine",different_base=None):
    add_fact(obst_name+"("+str(i)+","+str(j)+")",different_base)

def grab_message(i,j,msgs,text,different_base=None):
    if msgs.at(i,j)[0]==1:
       print ("I can add  message")
       add_fact(msgs.at(i,j)[1],different_base)
```

```
        print(text.at(i,j)[1])

text = Propety(8, "message")
msgs = Propety(8, "message")
obst = Propety(8, "mine")
visited = Propety(8, "visited")




def build_dict_from_messages():
   dict=[]
   for i in range(1,msgs.dim+1):
       for j in range(1, msgs.dim+1):
           if msgs.at(i,j)[0]==1:
            dict.append({"x":j,"y":i,"msg":msgs.at(i,j)[1]})
   return dict

def build_dict_from_texts():
    dict = []
    for i in range(1, text.dim + 1):
        for j in range(1, text.dim + 1):
            if text.at(i, j)[0] == 1:
                dict.append({"x": j, "y": i, "text": text.at(i, j)[1]})
    return dict
def build_dict_from_obst():
    dict = []
    for i in range(1, obst.dim + 1):
        for j in range(1, obst.dim + 1):
            if obst.at(i, j)[0] == 1:
                dict.append({"x": j, "y": i})
    return dict

def set_visited_fol(i,j,different_base=None):
    remove_fact("-visited("+str(i)+","+str(j)+")",different_base)
    add_fact("visited("+str(i)+","+str(j)+")",different_base)

def check_win(different_base=None):
    print("check win")
    set_goal("Win")
    return check_usingMace4("mineFOL",different_base)


def check_lose(different_base=None):
    print("check lost")
    set_goal("-lose")
    return  not check_usingMace4("mineFOL",different_base)


def generate_random_bomb(d=8,n=4):
    for i in range(n):
        x= int(1+random.random()*d)
        y = int(1 + random.random() * d)
        obst.set_state(x,y,1)
```

```
#code only for MineFol


from commonsFOL import *

assunptions_ruler=[]




def setup_assumptions_mine_fol(n=9):
    add_option1("domain_size,"+str(n))
    add_option2("arithmetic")
    add_fact("mine(x,y) | safe(x,y)")
    add_rule("mine(x,y)->-safe(x,y)")
    add_fact("safe(1,1)")


def setup_assumptions_rule_maker(n=9):
    add_rule(" exists x  exists y  (x>=1  & y>=1 & -visited(x,y) & -mine(x,y))  ->-Win",assunptions_r
    add_rule(" exists x exists y (x>=1 & y>=1  & (visited(x,y) & mine(x,y)) ) -> lose",assunptions_ru
    for i in range(1, n):
        for j in range(1, n):
          if  (not (i==1) ) or  (not (j==1)):
            add_fact("-visited(" + str(i) + "," + str(j) + ")",assunptions_ruler)
          else:
            add_fact("visited(" + str(i) + "," + str(j) + ")",assunptions_ruler)

    for i in range(1,n):
        for j in range(1, n):
            if obst.at(i,j)[0]==1:
                add_fact("mine(" + str(i) + "," + str(j) + ")",assunptions_ruler)
            else:
                add_fact("-mine(" + str(i) + "," + str(j) + ")",assunptions_ruler)



def reset_page(n=9):
    print("reset")
    remove_assumptions()
    remove_assumptions(assunptions_ruler)
    remove_options()
    remove_goals()
    setup_assumptions_mine_fol(n)
    setup_assumptions_rule_maker(n)
    global visited
    visited=Propety(n, "visited")



#Code only for Escape Room Game
from commonsFOL import *
from prover9_lib import remove_fact_re
```

```
keyx = 0
keyy = 0
doorx = 0
doory = 0

assunptions_ruler = []

import re


def get_keyx():
    return keyx

def get_keyy():
    return keyy

def get_doorx():
    return doorx
def get_doory():
    return doory

def load_key_door(path):

    print("key-door loaded")
    f = open(path)
    line = f.readline()
    result = re.search("(\d+) (\d+) (\d+) (\d+)", line)
    if not result == None:
        global keyy, keyx, doorx, doory
        print("found pattern")
        keyx = result.group(1)
        keyy = result.group(2)
        doorx = result.group(3)
        doory = result.group(4)


def setup_assumptions_escape_fol(n=9):
    add_option1("domain_size," + str(n))
    add_option2("arithmetic")
    add_fact("mine(x,y) | safe(x,y)")

    add_rule("mine(x,y)->-safe(x,y)")
    add_fact("safe(1,1)")


def setup_assumptions_rule_maker(n=9):
    add_rule("exists x exists y (x>=1 & y>=1  &  key(x,y) & visited(x,y) ) <-> key_found ", assunptio
    add_rule("exists x (exists y (x>=1 & y>=1  &  visited(x,y) & door(x,y) )) <-> door_found", assunp
    add_rule("door(x,y) & door(z,w) -> x=z & y=w", assunptions_ruler)
    add_rule("key(x,y) & key(z,w) -> x=z & y=w", assunptions_ruler)
    add_rule("exists x exists y (x>=1 & y>=1  & (visited(x,y) & mine(x,y)) )-> lose", assunptions_rul
    add_rule("door_found & key_found <-> Win", assunptions_ruler)

    add_fact("key(" + str(keyy) + "," + str(keyx) + ")", assunptions_ruler)
    add_fact("door(" + str(doory) + "," + str(doorx) + ")", assunptions_ruler)
```

```
    # add_fact(" -(x="+str(doory)+") | -(y="+str(doorx)+")" +"-> -door(x,y)",assunptions_ruler)
     #add_fact(" -(x=" + str(keyy) + ") | -(y=" + str(keyx) + ")" + "-> -key(x,y)",assunptions_ruler)
     for i in range(1, n):
         for j in range(1, n):
             add_fact("-visited(" + str(i) + "," + str(j) + ")", assunptions_ruler)

     for i in range(1, n):
         for j in range(1, n):
             if obst.at(i, j)[0] == 1:
                 add_fact("mine(" + str(i) + "," + str(j) + ")", assunptions_ruler)
             else:
                 add_fact("-mine(" + str(i) + "," + str(j) + ")", assunptions_ruler)

def set_key(i, j):
    add_fact("key(" + str(i) + "," + str(j) + ")")


def set_door(i, j):
    add_fact("door(" + str(i) + "," + str(j) + ")")


def reset_page(n=9):
    print("reset")
    remove_assumptions()
    remove_assumptions(assunptions_ruler)
    remove_options()
    remove_goals()
    setup_assumptions_escape_fol(n)
    setup_assumptions_rule_maker(n)
    global visited
    visited = Propety(n, "visited")


def check_win(different_base=None):
    print( "check win")
    set_goal("-Win")
    return not check_usingMace4("EscapeJail", different_base)


#Code Web Server MineFOL:
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
import time

hostName = "localhost"
serverPort = 8080
from FOL import *
import re
import json
import os
class MyServer(BaseHTTPRequestHandler):
    def do_GET(self):
        print("path of request is:" ,self.path)
        result= re.search("/messages",self.path)
        if(not result==None):
            map1={"fol":build_dict_from_messages(),
                  "texts": build_dict_from_texts()}
```

```
            reset_page()
            json_string =json.dumps(map1)
            # print json_string
            self.send_response(200)
            self.send_header("Content-Type", "application/json")
            self.send_header('Access-Control-Allow-Origin', '*')
            self.send_header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PUT, PATCH, DELETE')
            self.send_header('Access-Control-Allow-Headers',"Content-Type")
            self.end_headers()
            self.wfile.write(json_string.encode(encoding='utf_8'))
            return

        else:
            result = re.search("/bombs", self.path)
            if (not result == None):
                map1 = {"obstacol":build_dict_from_obst()}
                json_string=json.dumps(map1)
                # print json_string
                self.send_response(200)
                self.send_header("Content-type", "application/json")
                self.send_header('Access-Control-Allow-Origin', "*")
                self.send_header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PUT, PATCH, DELETE
                self.end_headers()
                self.wfile.write(json_string.encode(encoding='utf_8'))

                return
            else:
             result= re.search("/safe/i=(\d+),j=(\d+)", self.path)
             if not result==None:
               i=int(result.group(1))
               j=int(result.group(2))
               print("search safe at:",i,j)
               ok=False
               safe=check_safe(i,j)

               if not safe:
                 mine=check_prop(i,j)
                 if mine==False:
                      ok=True
               else:
                ok= True
               if ok:
                   map1 = {"safe":"true"}
                   json_string = json.dumps(map1)
                   print json_string
                   self.send_response(200)
                   self.send_header("Content-Type", "application/json")
                   self.send_header('Access-Control-Allow-Origin', '*')
                   self.send_header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PUT, PATCH, DEL
                   self.send_header('Access-Control-Allow-Headers', "Content-Type")
                   self.end_headers()
                   self.wfile.write(json_string.encode(encoding='utf_8'))
                   return
               else :
                   map1 = {"safe": "false"}
                   json_string = json.dumps(map1)
```

```
            print json_string
            self.send_response(200)
            self.send_header("Content-Type", "application/json")
            self.send_header('Access-Control-Allow-Origin', '*')
            self.send_header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PUT, PATCH, DEL
            self.send_header('Access-Control-Allow-Headers', "Content-Type")
            self.end_headers()
            self.wfile.write(json_string.encode(encoding='utf_8'))
            return

        else:
            result = re.search("/message/i=(\d+),j=(\d+)", self.path)
            if not result == None:
                i = result.group(1)
                j = result.group(2)
                print("add message at:",i,j)
                grab_message(int(i),int(j),msgs,text)
                self.send_response(200)
                self.send_header("Content-Type", "application/json")
                self.send_header('Access-Control-Allow-Origin', '*')
                self.send_header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PUT, PATCH, DE
                self.send_header('Access-Control-Allow-Headers', "Content-Type")
                self.end_headers()
                map1 = {"message_position": "(" + str(i) + "," + str(j) + ")"}
                json_string = json.dumps(map1)
                self.wfile.write(json_string.encode(encoding='utf_8'))
                return

            else:
                result = re.search("/visited/i=(\d+),j=(\d+)", self.path)
                if not result == None:
                    i = int(result.group(1))
                    j = int(result.group(2))
                    print("visited position:", i, j)
                    visited.set_state(i,j,1)
                    set_visited_fol(i,j,assunptions_ruler)
                    # print("visited:",visited.locations)
                    self.send_response(200)
                    self.send_header("Content-Type", "application/json")
                    self.send_header('Access-Control-Allow-Origin', '*')
                    self.send_header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PUT, PATCH
                    self.send_header('Access-Control-Allow-Headers', "Content-Type")
                    self.end_headers()
                    map1 = {"visited_position": "(" +str(i)+","+str(j)+")"}
                    json_string = json.dumps(map1)
                    self.wfile.write(json_string.encode(encoding='utf_8'))
                    return

                else:
                    result = re.search("/won", self.path)
                    if not result == None:
                        self.send_response(200)
                        self.send_header("Content-Type", "application/json")
                        self.send_header('Access-Control-Allow-Origin', '*')
                        self.send_header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PUT, P
                        self.send_header('Access-Control-Allow-Headers', "Content-Type")
```

```
            self.end_headers()

            if check_win(assunptions_ruler):

                map1 = {"win": "true"}
                json_string = json.dumps(map1)
                self.wfile.write(json_string.encode(encoding='utf_8'))
                return
            else:
                map1 = {"win": "false"}
                json_string = json.dumps(map1)
                self.wfile.write(json_string.encode(encoding='utf_8'))
                return
        else:
            result = re.search("/lose", self.path)
            if not result == None:
                self.send_response(200)
                self.send_header("Content-Type", "application/json")
                self.send_header('Access-Control-Allow-Origin', '*')
                self.send_header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PU
                self.send_header('Access-Control-Allow-Headers', "Content-Type")
                self.end_headers()
                if check_lose(assunptions_ruler):
                  map1 = {"lost": "true"}
                  json_string = json.dumps(map1)
                  self.wfile.write(json_string.encode(encoding='utf_8'))
                  return
                else:
                  map1 = {"lost": "false"}
                  json_string = json.dumps(map1)
                  self.wfile.write(json_string.encode(encoding='utf_8'))
                  return
            else:
                result = re.search("/set_mine/i=(\d+),j=(\d+)", self.path)
                if not result == None:
                    i = int(result.group(1))
                    j = int(result.group(2))
                    print("mine position:", i, j)
                    visited.set_state(i, j, 1)
                    set_obst(i, j)
                    self.send_response(200)
                    self.send_header("Content-Type", "application/json")
                    self.send_header('Access-Control-Allow-Origin', '*')
                    self.send_header('Access-Control-Allow-Methods',
                                     'HEAD, GET, POST, PUT, PATCH, DELETE')
                    self.send_header('Access-Control-Allow-Headers', "Content-Type")
                    self.end_headers()
                    map1 = {"mine_position": "(" + str(i) + "," + str(j) + ")"}
                    json_string = json.dumps(map1)
                    self.wfile.write(json_string.encode(encoding='utf_8'))
                    return
                else:
                   result = re.search("/reset", self.path)
                   if not result == None:
                    reset_game_new_game()
                    self.send_response(200)
```

```
                                        self.send_header("Content-Type", "application/json")
                                        self.send_header('Access-Control-Allow-Origin', '*')
                                        self.send_header('Access-Control-Allow-Methods',
                                                         'HEAD, GET, POST, PUT, PATCH, DELETE')
                                        self.send_header('Access-Control-Allow-Headers', "Content-Type")
                                        self.end_headers()
                                        map1 = {"reseted":"ok"}
                                        json_string = json.dumps(map1)
                                        self.wfile.write(json_string.encode(encoding='utf_8'))
                                        return

        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write(bytes("<p>Request: %s</p>" % self.path))
        self.wfile.write(bytes("<html><head><title>https://pythonbasics.org</title></head>"))
        self.wfile.write(bytes("<p>Request: %s</p>" % self.path))
        self.wfile.write(bytes("<body>"))
        self.wfile.write(bytes("<p>This is an example web server.</p>"))
        self.wfile.write(bytes("</body></html>"))

from sys import argv
import os
import random









def process_argv():

  for option in argv:
     if option=="-b":
         path= argv[argv.index(option)+1]
         load_obstacles(path,obst)
     if option=="-m":
         path= argv[argv.index(option)+1]
         load_messages(path,msgs, text)
     if option=="-v":
         version= argv[argv.index(option)+1]
         ver_result=choce_random_game_version(version)
         load_messages(ver_result[1],msgs, text)
         load_obstacles(ver_result[0],obst)



def choce_random_game_version(ver=None):

 file_dir_path=os.path.dirname(os.path.realpath("__file__"))
 g_dir="/games_configurations"
 files1=os.listdir("."+g_dir)
 #bomb
```

```
 r = re.compile("bombs(.*)")
 files=filter(r.match,files1)
 print(files)
 #versions of games
 versions=map(r.search,files)
 versions2=[]
 for v in versions:
  versions2.append(v.group(1))
 print(versions2)

  #choose a random version
 paths=[]
 if ver==None:
   version=versions2[int(random.random()*(len(versions2)-1))]
   print("Your chosen version is:",version)
   paths=[file_dir_path+g_dir+"/bombs"+version,file_dir_path+g_dir+"/messages"+version]
 else:
   paths=[file_dir_path+g_dir+"/bombs"+ver,file_dir_path+g_dir+"/messages"+ver]
   print("Your chosen version is:",ver)

 print(paths)
 return paths


def reset_game_new_game():
    version=choce_random_game_version()
    load_messages(version[1], msgs, text)
    load_obstacles(version[0], obst)

    process_argv()
    setup_assumptions_mine_fol()
    setup_assumptions_rule_maker()




if __name__ == "__main__":
    webServer = HTTPServer((hostName, serverPort), MyServer)
    print("Server started http://%s:%s" % (hostName, serverPort))
    #random setup for game
    reset_game_new_game()



#Code Server Escape Jail

    else:
                                    result = re.search("/key", self.path)
                                    if not result == None:
                                        print("key sending:")
                                        self.send_response(200)
                                        self.send_header("Content-Type", "application/json")
                                        self.send_header('Access-Control-Allow-Origin', '*')
                                        self.send_header('Access-Control-Allow-Methods',
                                                         'HEAD, GET, POST, PUT, PATCH, DELETE')
```

```
                                        self.send_header('Access-Control-Allow-Headers', "Content-Typ
                                        self.end_headers()
                                        map1 = {"y":get_keyy(),"x":get_keyx()}
                                        json_string = json.dumps(map1)
                                        self.wfile.write(json_string.encode(encoding='utf_8'))
                                        return
                                    else:

                                        result = re.search("/door", self.path)
                                        if not result == None:
                                            print("door sending:")
                                            self.send_response(200)
                                            self.send_header("Content-Type", "application/json")
                                            self.send_header('Access-Control-Allow-Origin', '*')
                                            self.send_header('Access-Control-Allow-Methods',
                                                             'HEAD, GET, POST, PUT, PATCH, DELETE')
                                            self.send_header('Access-Control-Allow-Headers', "Content
                                            self.end_headers()
                                            map1 = {"y": get_doory(), "x": get_doorx()}
                                            json_string = json.dumps(map1)
                                            self.wfile.write(json_string.encode(encoding='utf_8'))
                                            return
    load_key_door("key_door")
    setup_assumptions_escape_fol()
from sys import argv
def process_argv():

  for option in argv:
     if option=="-b":
           path= argv[argv.index(option)+1]
           load_obstacles(path,obst)
     if option=="-m":
         path= argv[argv.index(option)+1]
         load_messages(path,msgs, text)
     if option=="-k_d":
         path= argv[argv.index(option)+1]
         load_key_door(path)
     if option=="-v":
         version= argv[argv.index(option)+1]
         ver_result=choce_random_game_version(version)
         load_messages(ver_result[1],msgs, text)
         load_obstacles(ver_result[0],obst)
         load_key_door(ver_result[2])




def choce_random_game_version(ver=None):

 file_dir_path=os.path.dirname(os.path.realpath("__file__"))
 g_dir="/games_configurations"
```

```
 files1=os.listdir("."+g_dir)
 #bomb
 r = re.compile("bombs(.*)")
 files=filter(r.match,files1)
 print(files)
 #versions of games
 versions=map(r.search,files)
 versions2=[]
 for v in versions:
  versions2.append(v.group(1))
 print(versions2)

  #choose a random version
 paths=[]
 if ver==None:
   version=versions2[int(random.random()*(len(versions2)-1))]
   print("Your chosen version is:",version)
   paths=[file_dir_path+g_dir+"/bombs"+version,file_dir_path+g_dir+"/messages"+version,file_dir_path+
 else:
   paths=[file_dir_path+g_dir+"/bombs"+ver,file_dir_path+g_dir+"/messages"+ver,file_dir_path+g_dir+"/
   print("Your chosen version is:",ver)
 print(paths)
 return paths



def reset_game_new_game():
    version=choce_random_game_version()
    load_messages(version[1], msgs, text)
    load_obstacles(version[0], obst)
    load_key_door(version[2])


    process_argv()

    setup_assumptions_escape_fol()
    setup_assumptions_rule_maker()


if __name__ == "__main__":
    webServer = HTTPServer((hostName, serverPort), MyServer)
    print("Server started http://%s:%s" % (hostName, serverPort))
    reset_game_new_game()

    print("key-door" ,get_keyx() ,get_keyy(),get_doorx(),get_doory())

    #run_game(8, visited, obst, msgs, text)



#game autogenerator Code


from propety import Propety
import random
```

```
def generate_random_bomb(obst,version,n=4,d=8):
    f= open("./games_configurations/bombs"+version,"w+")
    print("thisis my n",n)
    nr_bomb=n
    while nr_bomb>0:

        x= int(1+random.random()*d)
        y = int(1 + random.random() * d)
        if (not x==1) or (not y==1) & obst.at(x,y)[0]==0:
            obst.set_state(x,y,1)

            f.write(str(x)+" "+str(y)+"\n")
            nr_bomb-=1

def valid_position(i,j,obst,messages):
    if obst.at(i,j)[0]==0 and messages.at(i,j)[0]==0:
        return True
    else:
        return False
def build_valid_positions_list(obst,messages,n=9):
     list_valid=[]
     for i in range(1,n):
         for j in range(1, n):
            if valid_position(i,j,obst,messages):
                list_valid.append((i,j))
     return list_valid


def choice_message_position(obst,messages,n=9):
    index_try=0
    list=build_valid_positions_list(obst,messages,n=9)
    limit_tries = 10 * len(list)
    if len(list)>0:
     i1=int(1+random.random()*(len(list)-2))
     elm=list[i1]
     list.pop(i1)
     return elm

    else:
        return None




def build_random_fol_valid_messages(obst,version,nr_messages_to_build=10):
 f= open("./games_configurations/messages"+version,"w+")
 messages=Propety(obst.dim,"messages")
 texts=Propety(obst.dim,"texts")
 print("build ",nr_messages_to_build," messages")

 while nr_messages_to_build >0 :
```

```
choice = int(1+random.random() * 18)
position=choice_message_position(obst,messages,obst.dim)
if not position ==None:
   if choice == 1:
        #give the location of a mine
        bombs=[]
        for i in range(1,obst.dim+1):
           for j in range(1,obst.dim+1):
              if obst.at(i,j)[0]==1:
                 bombs.append((i,j))
        if len(bombs)>0:
         index=int(random.random()*(len(bombs)-1))
         messages.at(position[0],position[1])[0]=1
         texts.at(position[0],position[1])[0]=1
         f.write(str(position[0])+" "+ str(position[1])+" "
         +"mine("+str(bombs[index][0]) +","+str(bombs[index][1])+")."
          +" mine on location "+"("+str(bombs[index][0]) +","+str(bombs[index][1])+")\n")
         nr_messages_to_build-=1
        continue
   if choice == 2:
            #give the location of a free position
         no_bombs=[]
         for i in range(1,obst.dim+1):
           for j in range(1,obst.dim+1):
              if obst.at(i,i)[0]==0:
                 no_bombs.append((i,j))
         if len(no_bombs)>0:
          index=int(random.random()*(len(no_bombs)-1))
          messages.at(position[0],position[1])[0]=1
          texts.at(position[0],position[1])[0]=1
          f.write(str(position[0])+" "+ str(position[1])+" "
          +"-mine("+str(no_bombs[index][0]) +","+str(no_bombs[index][1])+")."
          +"no mine on location "+"("+str(no_bombs[index][0]) +","+str(no_bombs[index][1])+")\n"
          nr_messages_to_build-=1
         continue
   if choice == 3:

       # check for a free main diagonal
       free_d=True
       for i in range(1,obst.dim+1):
         if obst.at(i,i)[0]==1:
           free_d=False

       if free_d:
          messages.at(position[0],position[1])[0]=1
          texts.at(position[0],position[1])[0]=1
          f.write(str(position[0])+" "+ str(position[0])+" "
          +"-(exists x (x>=1 & mine(x,x)))."
          +  "no mine on the main diagonal "+"\n")
          nr_messages_to_build-=1
          continue
   if choice == 4:
       # check free second diagonal
       free_d=True
       for i in range(1,obst.dim+1):
         if obst.at(i,obst.dim-1-i)[0]==1:
```

```
            free_d=False
        if free_d:
           messages.at(position[0],position[1])[0]=1
           texts.at(position[0],position[1])[0]=1
           f.write(str(position[0])+" "+ str(position[1])+" "
           +"-(exists x (x>=1 &  mine(x,"+str(obst.dim)+"-x)))."
           +" no mine on the second diagonal "+"\n")
           nr_messages_to_build-=1
           continue
    if choice == 5:
        # check for a free row
        free_lines=[]
        for i in range(1,obst.dim+1):
            free_line=True
            for j in range(1,obst.dim+1):
                if obst.at(i,j)[0]==1:
                    free_line=False
            if free_line:
                   free_lines.append(i)
        if len(free_lines)>0 :
           index=int(random.random()*(len(free_lines)-1))
           messages.at(position[0],position[1])[0]=1
           texts.at(position[0],position[1])[0]=1
           f.write(
            str(position[0])+" "+ str(position[1])+" "+
           "-(exists x (x>=1 &  mine("+str(free_lines[index])+",x)))."+
           " no mine on the row "+str(free_lines[index])+"\n")
           nr_messages_to_build-=1
        continue
    if choice == 6:
          #check for free column
        free_columns=[]
        for i in range(1,obst.dim+1):
            free_column=True
            for j in range(1,obst.dim+1):
                if obst.at(j,i)[0]==1:
                    free_column=False
            if free_column:
                   free_columns.append(i)
        if len(free_columns)>0 :
           index=int(random.random()*(len(free_columns)-1))
           messages.at(position[0],position[1])[0]=1
           texts.at(position[0],position[1])[0]=1
           f.write(
           str(position[0])+" "+ str(position[1])+" "+
           "-(exists x (x>=1 & mine(x,"+str(free_columns[index])+"))))."+
           " no mine on column "+str(free_columns[index])+"\n")
           nr_messages_to_build-=1
        continue

    if choice == 7:
        # check general simetry x,z =z,x 1.
         simetry1=True
         for i in range(1,obst.dim+1):
            for j in range(1, obst.dim+1):
                if not obst.at(i,j)==obst.at(j,i):
```

```
                simetry1=False
        if simetry1:
          messages.at(position[0],position[1])[0]=1
          texts.at(position[0],position[1])[0]=1
          f.write(
          str(position[0])+" "+ str(position[1])+" "+
          "mine(x,y)->mine(y,x)."+
          "mines are disposed simetric by the main diagonal"+"\n")
          nr_messages_to_build-=1
        continue

    if choice == 8:
        # check general simetry x,8-x,  1.
        simetry1=True
        for i in range(1,obst.dim+1):
           for j in range(1,obst.dim+1):
               if not obst.at(i,j)==obst.at(obst.dim-j,obst.dim-i):
                  simetry1=False
        if simetry1:
          messages.at(position[0],position[1])[0]=1
          texts.at(position[0],position[1])[0]=1
          f.write(
          str(position[0])+" "+ str(position[1])+" "+
          "mine(x,y)->mine(8-y,8-x)."+
          "mines are disposed simetric by the second diagonal"+"\n")
          nr_messages_to_build-=1
        continue
    if choice == 9:
       #check same line positioning
       same_lines=[]
       for i in range(1,obst.dim+1):
            for j in range(1, obst.dim+1):

             same_line=True
             for k in range(1,obst.dim+1):
               if i==j or not obst.at(i,k)==obst.at(j,k)  :
                 same_line=False
             if same_line==True:
                same_lines.append((i,j))
       if len(same_lines)>0:
           index=int(random.random()*(len(same_lines)-1))
           messages.at(position[0],position[1])[0]=1
           texts.at(position[0],position[1])[0]=1
           f.write(
           str(position[0])+" "+ str(position[1])+" "+
           "exists x (x>=1 & mine("+str(same_lines[index][0])+",x)->mine("+str(same_lines[index][
           "if there is a mine on the line "+str(same_lines[index][0])+" there is a bomb on the l
           nr_messages_to_build-=1
       continue
    if choice == 10:
          #check same column positioning
       same_columns=[]
       for i in range(1,obst.dim+1):
            for j in range(1, obst.dim+1):

             same_column=True
```

```
            for k in range(1, obst.dim+1):
              if i==j or not obst.at(k,i)==obst.at(k,j):
                same_column=False
            if same_column==True:
               same_columns.append((i,j))
    if len(same_columns)>0:
        index=int(random.random()*(len(same_columns)-1))
        messages.at(position[0],position[1])[0]=1
        texts.at(position[0],position[1])[0]=1
        f.write(
        str(position[0])+" "+ str(position[1])+" "+
        "exists x (x>=1 & mine(x,"+str(same_columns[index][0])+")->mine(x,"+str(same_columns[i
        "if there is a mine on the column "+str(same_columns[index][0])+" there is a bomb on t
        nr_messages_to_build-=1
    continue
    if choice == 11:
     # check general simetry x,z =z,x 1.
      simetrics=[]
      for i in range(1,obst.dim+1):
         for j in range(1, obst.dim+1):
           simetry1=True
           for k  in range(1,obst.dim+1):
             if not obst.at(i,k)==obst.at(k,j):
                simetry1=False
           if simetry1:
             simetrics.append((i,j))

      if len(simetrics)>0:
        index_s=int(random.random()*(len(simetrics)+1))
        messages.at(position[0],position[1])[0]=1
        texts.at(position[0],position[1])[0]=1
        f.write(
        str(position[0])+" "+ str(position[1])+" "+
        "mine("+str(simetrics[index_s][0])+",y)<->mine(y,"+str(simetrics[index_s][1])+")."+
        "if there are mines on the row"+str(simetrics[index_s][0])+"there are mines on the  co
        nr_messages_to_build-=1
      continue

if choice == 12:
     # no bomb on a row lower than the bomb with the smallest index
      minim=obst.dim+2
      for i in range(1,obst.dim+1):
         for j in range(1, obst.dim+1):
              if obst.at(i,j)[0]==1:
                 if i<minim:
                    minim=i
      if  not minim<=1:
        messages.at(position[0],position[1])[0]=1
        texts.at(position[0],position[1])[0]=1
        f.write(
        str(position[0])+" "+ str(position[1])+" "+
        "exists y (y<"+str(minim)+"-> -mine(y,x))."
        "No mine has the row smaller than "+str(minim)+"\n")
        nr_messages_to_build-=1
      continue
if choice == 13:
```

```
    # no bomb on a column lower than the bomb with the smallest index
     minim=obst.dim+2
     for i in range(1,obst.dim+1):
        for j in range(1, obst.dim+1):
             if obst.at(i,j)[0]==1:
                if j<minim:
                   minim=j
     if  not minim<=1:
       messages.at(position[0],position[1])[0]=1
       texts.at(position[0],position[1])[0]=1
       f.write(
       str(position[0])+" "+ str(position[1])+" "+
       "exists x (x<"+str(minim)+"-> -mine(y,x))."
       "No mine has the column smaller than "+str(minim)+"\n")
       nr_messages_to_build-=1
     continue
if choice == 14:
    # no bomb on a row bigger than the bomb with the biggest index
     maxim=-1
     for i in range(1,obst.dim+1):
        for j in range(1, obst.dim+1):
             if obst.at(i,j)[0]==1:
                if i>maxim:
                   maxim=i
     if  not maxim>=obst.dim:
       messages.at(position[0],position[1])[0]=1
       texts.at(position[0],position[1])[0]=1
       f.write(
       str(position[0])+" "+ str(position[1])+" "+
       "exists y (y>"+str(maxim)+"-> -mine(y,x))."
       "No mine has the row bigger than "+str(maxim)+"\n")
       nr_messages_to_build-=1
     continue
if choice == 15:
    # no bomb on a column bigger than the bomb with the biggest index
     maxim=-1
     for i in range(1,obst.dim+1):
        for j in range(1, obst.dim+1):
             if obst.at(i,j)[0]==1:
                if j>maxim:
                   maxim=j
     if  not maxim>=obst.dim:
       messages.at(position[0],position[1])[0]=1
       texts.at(position[0],position[1])[0]=1
       f.write(
       str(position[0])+" "+ str(position[1])+" "+
       "exists x (x<"+str(maxim)+"-> -mine(y,x))."
       "No mine has the column smaller than "+str(maxim)+"\n")
       nr_messages_to_build-=1
     continue
if choice == 16:
    # share the position of two boms
     chosen_first=False
     chosen_second=False
     i1=[0,0]
     i2=[0,0]
```

```
            for i in range(1,obst.dim+1):
               for j in range(1, obst.dim+1):
                    if obst.at(i,j)[0]==1:
                        if chosen_first==False:
                          i1[0]=i
                          i1[1]=j
                          chosen_first=True
                        else:
                         if chosen_second==False:
                          i2[0]=i
                          i2[1]=j
                          chosen_second=True
            f.write(
            str(position[0])+" "+ str(position[1])+" "+
            "mine("+str(i1[0])+","+str(i1[1])+")->mine("+str(i2[0])+","+str(i2[1])+")."+
            "if there is one mine on ("+str(i1[0]) +","+str(i1[1]) +") than there is one also on ("+
            nr_messages_to_build-=1
            continue
        if choice == 17:
            # a bomb on a line
            bombs=[]
            for i in range(1,obst.dim+1):
               for j in range(1, obst.dim+1):
                    if obst.at(i,j)[0]==1:
                     bombs.append((i,j))
            bomb= random.choice(bombs)
            f.write(
            str(position[0])+" "+ str(position[1])+" "+
            "existx x (x>=1 & mine("+str(bomb[0])+",x))."+
            "there is at least a mine on line "+str(bomb[0])+"\n")
            nr_messages_to_build-=1
            continue
        if choice == 18:
            # a bomb on a column
            bombs=[]
            for i in range(1,obst.dim+1):
               for j in range(1, obst.dim+1):
                    if obst.at(i,j)[0]==1:
                     bombs.append((i,j))
            bomb= random.choice(bombs)
            f.write(
            str(position[0])+" "+ str(position[1])+" "+
            "exists x (x>=1 & mine(x,"+str(bomb[1])+"))."+
            "there is at least a mine on column "+str(bomb[1])+"\n")
            nr_messages_to_build-=1
            continue
    else:
     f.close()
     return messages
 f.close()
 return messages

import sys

n=4
version="_3"
```

```
nr_m=10

def process_argv():
 global n,version,nr_m
 for a in sys.argv:
   if a=="-n_b":
       n=int(sys.argv[sys.argv.index(a)+1])
   if a=="-n_m":
       nr_m= int(sys.argv[sys.argv.index(a)+1])
   if a=="-v":
       version=sys.argv[sys.argv.index(a)+1]

if __name__ == "__main__":
  obst=Propety(8,"mines")
  process_argv()
  print "here",n
  generate_random_bomb(obst,version,n)
  build_random_fol_valid_messages(obst,version,nr_m)



//Code in javascript for web interface

//init_page1.js
const image = new Image(60,60);
image.src = "images/envelope.png";
const image2 = new Image(60,60);
image2.src = "images/bomb.png";
const image3 = new Image(60,60);
image3.src = "images/flag.png";

var bombs=[]
var messages=[]

var pressed=0
function showInstructions()
{
var pmes=document.getElementById("instructions")
if(pmes)
if(pressed==0)
{
pressed=1
pmes.style.display="block"
}
else
{
pressed=0
pmes.style.display="none"
}

}

async function get(){
return fetch("http://localhost:8080/messages",{method:"GET",mode:"cors"}).then((response) => response
  .then((data) => {
    console.log('Success:',data);
     messages=data['texts']
```

```
  })
  .catch((error) => {
    console.error('Error:', error);
  });




}

async function get_bombs()
{
await get()
return fetch("http://localhost:8080/bombs",{method:"GET",mode:"cors"}).then((response) => response.js
  .then((data) => {
    console.log('Success:',data);
     bombs=data["obstacol"]

  })
  .catch((error) => {
    console.error('Error:', error);
  });
}

get_bombs()


window.addEventListener("keydown", function(e) {
    if(["Space","ArrowUp","ArrowDown","ArrowLeft","ArrowRight"].indexOf(e.code) > -1) {
        e.preventDefault();
    }
}, false);

//Escape jail game init_page2.js


async function get_door()
{
await get_bombs()
return fetch("http://localhost:8085/door",{method:"GET",mode:"cors"}).then((response) => response.jso
  .then((data) => {
     console.log('Door of Success:',data);
     door=data

  })
  .catch((error) => {
    console.error('Error:', error);
  });
}


async function get_key()
{
```

```
await get_door()
return fetch("http://localhost:8085/key",{method:"GET",mode:"cors"}).then((response) => response.json
  .then((data) => {
    console.log('Success key:',data);
     key=data

  })
  .catch((error) => {
    console.error('Error:', error);
  });
}

get_key()

//MineFOL game_page1.js
console.log('messages:',messages);
var canvas= document.getElementById("board")
var context=canvas.getContext("2d")
var w=800
var h=735
var currentx=1
var currenty=1
var prevx
var prevy

var box=90
var drawableObjects=Array(9).fill(null).map( () => Array(9).fill(0))

   function drawboard()
   {

     context.lineWidth = 10;
     context.strokeStyle = "rgb(2,7,159)";
       for(var i=0;i<h;i+=box)
       for(var j=0;j<w;j+=box)
       {
         context.strokeRect(i+10, j+10, box,box);

       }

   }
   function drawImage(boxi,boxj)
{
   var boxh=(boxi-1)*90+10
   var boxw=(boxj-1)*90+10
   context.drawImage(image,boxh+15,boxw+10,60,60)

}
   function drawImage2(boxi,boxj)
{

   var boxh=(boxi-1)*90+10
   var boxw=(boxj-1)*90+10
   context.drawImage(image2,boxh+15,boxw+10,60,60)

}
```

```
   function drawImage3(boxi,boxj)
{
   var boxh=(boxi-1)*90+10
   var boxw=(boxj-1)*90+10
   context.drawImage(image3,boxh+15,boxw+10,60,60)

}
function check_message(cx,cy)
{
for( var m of messages)
{
if(m["y"]==cy && m["x"]==cx)
{

drawableObjects[cx][cy]=1
drawImage(cx,cy)

set_message(currentx,currenty,(data)=>console.log(data))

var pmes=document.getElementById("messages")
if(pmes)
pmes.textContent=m["text"]



}

}
}
function check_bomb(cx,cy)
{
for( var m of bombs)
{
if(m["y"]==cy && m["x"]==cx)
{
drawableObjects[cx][cy]=2
drawImage2(cx,cy)
return true


}

}
return false
}



function set_visited(cx,cy,do_stuff)
{

return fetch("http://localhost:8080/visited/i="+cy+",j="+cx,{method:"GET",mode:"cors"}).then((respons
  .then((data) => {


    do_stuff(data)
```

```
  })
  .catch((error) => {
    console.error('Error:', error);
  });
}
function set_message(cx,cy,do_stuff)
{

return fetch("http://localhost:8080/message/i="+cy+",j="+cx,{method:"GET",mode:"cors"}).then((respons
  .then((data) => {


    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });
}







async function check_win(cx,cy,do_stuff)
{

return fetch("http://localhost:8080/won",{method:"GET",mode:"cors"}).then((response) => response.json
  .then((data) => {


    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });
}


async  function check_lost(cx,cy,do_stuff)
{

return fetch("http://localhost:8080/lose",{method:"GET",mode:"cors"}).then((response) => response.jso
  .then((data) => {


    do_stuff(data)


  })
```

```
  .catch((error) => {
    console.error('Error:', error);
  });
}




async function inform_bomb_found(cx,cy, do_stuff)
{
return fetch("http://localhost:8080/set_mine/i="+cy+",j="+cx,{method:"GET",mode:"cors"}).then((respon
  .then((data) => {


    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });


}







async function askMace4(do_stuff)
{

return fetch("http://localhost:8080/safe/i="+currenty+",j="+currentx,{method:"GET",mode:"cors"}).then
  .then((data) => {


    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });

}









function drawCurrent()
{
```

```
 context.fillStyle ="#00FF00";
 context.fillRect((currentx-1)*90+15,(currenty-1)*90+15,80,80)

}
function drawPrevious()
{
 context.fillStyle = "#808080";
 context.fillRect((prevx-1)*90+15,(prevy-1)*90+15,80,80)

}


drawboard()







canvas.addEventListener('mousedown', function(e) {

   const rect = canvas.getBoundingClientRect()
    var x = e.clientX - rect.left
    var y = e.clientY - rect.top
    x=x/box
    y=y/box
    x=Math.trunc(x+1)
    y=Math.trunc(y+1)
    drawImage3(x,y)







})


function reset()
{
context.clearRect(0, 0, canvas.width, canvas.height);
drawboard()
currentx=1
currenty=1
var pmes=document.getElementById("messages")
if(pmes)
pmes.textContent=""
get_bombs()
}

function reload_page()
{
location.reload()

}
```

```
function show_answer()
{

context.clearRect(0,0,canvas.width,canvas.height)
drawboard()


for(var i=1 ;i<=8;i++)
for(var j=1 ;j<=8;j++)
{
 check_bomb(i,j)
  for( var m of messages)
{
if(m["y"]==i && m["x"]==j)
{

drawableObjects[i][j]=1
drawImage(i,j)

}

}
}

 drawCurrent()

}

function reset_game()
{
return fetch("http://localhost:8080/reset",{method:"GET",mode:"cors"}).then((response) => response.js
  .then((data) => {


    reload_page()


  })
  .catch((error) => {
    console.error('Error:', error);
  });

}









window.addEventListener('keydown',this.check,false);
```

```
async function check(e) {
   // alert(e.keyCode);

    prevx=currentx
    prevy=currenty


    if(e.keyCode==38)
    currenty-=1;

      if(e.keyCode==40)
     currenty+=1;


      if(e.keyCode==39)
      {
         currentx+=1;
      }
        if(e.keyCode==37)
      {
         currentx-=1;

      }

      if(currentx<1 || currentx>8)
      currentx=prevx
        if(currenty<1 || currenty>8)
      currenty=prevy




        askMace4( (data)=>
        {
        console.log("data:",data)
        if(data["safe"]=="false")
        {
         currentx=prevx
         currenty=prevy

        alert("Prover9 detactaed a bomb to the position you want to go, so we will block your moving"
        }
        else
        {



       check_bomb(currentx,currenty)
       check_message(currentx,currenty)


        set_visited(currentx,currenty,(data)=>console.log(data))

        if(check_bomb(currentx,currenty))
```

```
      {  inform_bomb_found(currentx,currenty,(data)=>console.log(data))

      }



      check_win(currentx,currenty,(data)=>{

       if(data["win"]=="true")
       {
       alert("You won , please play again!")
       reload_page()
       }

       } )


     check_lost(currentx,currenty,(data)=>{

       if(data["lost"]=="true")
       {
       alert("You lost, please try again!")
       reload_page()
       }

       } )

if(check_bomb(currentx,currenty))
      {

         currentx=1
         currenty=1

      }



     drawboard();



    if(drawableObjects[currentx][currenty]==0)
    {
      drawCurrent()
    }
    if(drawableObjects[prevx][prevy]==0)
    {
      drawPrevious()
    }
```

```
      }


    })

    }


//Escape jail game_page2.js
console.log('messages:',messages);
var canvas= document.getElementById("board")
var context=canvas.getContext("2d")
var w=800
var h=735
var currentx=1
var currenty=1
var prevx
var prevy

var box=90
var drawableObjects=Array(9).fill(null).map( () => Array(9).fill(0))

   function drawboard()
   {

     context.lineWidth = 10;
     context.strokeStyle = "rgb(128,128,128)";
       for(var i=0;i<h;i+=box)
       for(var j=0;j<w;j+=box)
       {
         context.strokeRect(i+10, j+10, box,box);

       }

   }
   function drawImage(boxi,boxj)
{
   var boxh=(boxi-1)*90+10
   var boxw=(boxj-1)*90+10
   context.drawImage(image,boxh+15,boxw+10,60,60)

}
   function drawImage2(boxi,boxj)

{
 var im_index=Math.trunc(Math.random()*3)
 console.log(im_index)
 image2.src=dangerSrc[im_index]



   var boxh=(boxi-1)*90+10
   var boxw=(boxj-1)*90+10
   context.drawImage(image2,boxh+15,boxw+10,60,60)
```

```
}
   function drawImage3(boxi,boxj)
{
   var boxh=(boxi-1)*90+10
   var boxw=(boxj-1)*90+10
   context.drawImage(image3,boxh+15,boxw+10,60,60)

}
   function drawImage4(boxi,boxj)
{
   var boxh=(boxi-1)*90+10
   var boxw=(boxj-1)*90+10
   context.drawImage(image4,boxh+15,boxw+10,60,60)

}
   function drawImage5(boxi,boxj)
{
   var boxh=(boxi-1)*90+10
   var boxw=(boxj-1)*90+10
   context.drawImage(image5,boxh+15,boxw+10,60,60)

}


function check_door(cx,cy)
{
if( door['x']==cx & door['y']==cy)
{
drawableObjects[cx][cy]=3
drawImage4(cx,cy)
}


}

function check_key(cx,cy)
{
if( key['x']==cx & key['y']==cy)
{
drawableObjects[cx][cy]=4
drawImage5(cx,cy)
}


}









function check_message(cx,cy)
{
for( var m of messages)
{
if(m["y"]==cy && m["x"]==cx)
```

```
{

drawableObjects[cx][cy]=1
drawImage(cx,cy)

set_message(currentx,currenty,(data)=>console.log(data))

var pmes=document.getElementById("messages")
if(pmes)
pmes.textContent=m["text"]



}

}
}
function check_bomb(cx,cy)
{
for( var m of bombs)
{
if(m["y"]==cy && m["x"]==cx)
{
drawableObjects[cx][cy]=2
drawImage2(cx,cy)
return true


}

}
return false
}


function set_visited(cx,cy,do_stuff)
{

return fetch("http://localhost:8085/visited/i="+cy+",j="+cx,{method:"GET",mode:"cors"}).then((respons
  .then((data) => {


    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });
}
function set_message(cx,cy,do_stuff)
{

return fetch("http://localhost:8085/message/i="+cy+",j="+cx,{method:"GET",mode:"cors"}).then((respons
  .then((data) => {
```

```
    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });
}




async function check_win(cx,cy,do_stuff)
{

return fetch("http://localhost:8085/won",{method:"GET",mode:"cors"}).then((response) => response.json
  .then((data) => {


   do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });
}


async  function check_lost(cx,cy,do_stuff)
{

return fetch("http://localhost:8085/lose",{method:"GET",mode:"cors"}).then((response) => response.jso
  .then((data) => {


    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });
}




async function inform_bomb_found(cx,cy, do_stuff)
{
return fetch("http://localhost:8085/set_mine/i="+cy+",j="+cx,{method:"GET",mode:"cors"}).then((respon
  .then((data) => {
```

```
    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });


}







async function askMace4(do_stuff)
{

return fetch("http://localhost:8085/safe/i="+currenty+",j="+currentx,{method:"GET",mode:"cors"}).then
  .then((data) => {


    do_stuff(data)


  })
  .catch((error) => {
    console.error('Error:', error);
  });

}









function drawCurrent()
{
 context.fillStyle = "#FFA500";
 context.fillRect((currentx-1)*90+15,(currenty-1)*90+15,80,80)

}
function drawPrevious()
{
 context.fillStyle = "#808080";
 context.fillRect((prevx-1)*90+15,(prevy-1)*90+15,80,80)

}


drawboard()
```

```
function reload_page()
{
location.reload()

}

function reset()
{
context.clearRect(0, 0, canvas.width, canvas.height);
drawboard()
currentx=1
currenty=1
var pmes=document.getElementById("messages")
if(pmes)
pmes.textContent=""
get_key()

}

function show_answer()
{

context.clearRect(0,0,canvas.width,canvas.height)
drawboard()


for(var i=1 ;i<=8;i++)
for(var j=1 ;j<=8;j++)
{
 check_bomb(i,j)
 check_door(i,j)
 check_key(i,j)
  for( var m of messages)
{
if(m["y"]==i && m["x"]==j)
{

drawableObjects[i][j]=1
drawImage(i,j)

}

}
}

 drawCurrent()

}


function reset_game()
{
return fetch("http://localhost:8085/reset",{method:"GET",mode:"cors"}).then((response) => response.js
  .then((data) => {
```

```
    reload_page()


  })
  .catch((error) => {
    console.error('Error:', error);
  });

}



canvas.addEventListener('mousedown', function(e) {

   const rect = canvas.getBoundingClientRect()
    var x = e.clientX - rect.left
    var y = e.clientY - rect.top
    x=x/box
    y=y/box
    x=Math.trunc(x+1)
    y=Math.trunc(y+1)
    drawImage3(x,y)







})


window.addEventListener('keydown',this.check,false);




async function check(e) {
   // alert(e.keyCode);

    prevx=currentx
    prevy=currenty


    if(e.keyCode==38)
    currenty-=1;

      if(e.keyCode==40)
     currenty+=1;


      if(e.keyCode==39)
      {
         currentx+=1;
```

```
}
  if(e.keyCode==37)
{
   currentx-=1;

}

if(currentx<1 || currentx>8)
currentx=prevx
  if(currenty<1 || currenty>8)
currenty=prevy




  askMace4( (data)=>
  {
  console.log("data:",data)
  if(data["safe"]=="false")
  {
   currentx=prevx
   currenty=prevy

  alert("Mace4 detactaed a bomb to the position you want to go, so we will block your moving")
  }
  else
  {


  check_door(currentx,currenty)
  check_key(currentx,currenty)
  check_bomb(currentx,currenty)
  check_message(currentx,currenty)


  set_visited(currentx,currenty,(data)=>console.log(data))


  if(check_bomb(currentx,currenty))
  {  inform_bomb_found(currentx,currenty,(data)=>console.log(data))

  }

   check_win(currentx,currenty,(data)=>{

   if(data["win"]=="true")
   {
   alert("You won , please play again!")
   reload_page()
   }else
   {

   check_lost(currentx,currenty,(data)=>{

   if(data["lost"]=="true")
   {
```

```
      alert("You lost, please try again!")
      reload_page()
      }

      } )
      }

      } )


if(check_bomb(currentx,currenty))
     {

        currentx=1
        currenty=1

     }



  // if(drawableObjects[prevx][prevy]==0)
    //context.clearRect((prevx-1)*box+10, (prevy-1)*box+10, box, box);

    drawboard();

   if(drawableObjects[currentx][currenty]==0)
   {
     drawCurrent()
   }

   if(drawableObjects[prevx][prevy]==0)
   {
     drawPrevious()
   }







   }


 })

 }
```