Tugas Besar 2:
Membuat Agen Minesweeper Berdasarkan *Knowledge Based System*

Ditujukan untuk memenuhi Tugas Mata Kuliah IF3170 Intelegensi Buatan

Disusun Oleh
Jundullah - 13518027
Rehan Adi Satrya - 13518061
Putri Nadia Salsabila - 13518094
Andjani Kiranadewi - 13518109

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2020

Introduction

Tugas Besar 2 pada kuliah Intelegensi Buatan ini bertujuan agar peserta kuliah mengimplementasikan KBS untuk membuat agen minesweeper. Minesweeper adalah game yang hanya dimainkan oleh satu pemain, dan tujuan game ini adalah membersihkan lahan tanpa mengenai bom. Rumusan masalah yang perlu diselesaikan oleh mahasiswa adalah:

1. Tahapan pembuatan aplikasi
2. Repository dokumentasi
3. User manual
4. Proses updating dan inferencing atas fakta yang terlibat

Tahapan Pembuatan Aplikasi

Tahapan yang dilakukan untuk pembuatan Minesweeper:

1. Menyiapkan program yang akan di run di CLIPS
    a. Menyiapkan kasus yang akan dibuat expert systemnya
    b. Menyiapkan program python yang akan dimainkan
    c. Menyiapkan deftemplate dan deffacts menggunakan kasus dan program python yang sudah dibuat
    d. Mendapatkan informasi dari expert tentang jalur yang harus ditempuh oleh sistem untuk menyelesaikan kasus yang sudah dibuat
    e. Dari informasi yang didapat dari expert akan dirancang menjadi kumpulan rule
    f. Merapikan program dengan deftemplate agar program lebih terstruktur
2. Menguji program
    a. Membuka file python
    b. Menjalankan program dengan perintah (run)
3. Program berhasil dijalankan
    a. Memeriksa hasil inferensi expert system yang dibuat
    b. Mengulang proses pembuatan dengan kasus yang berbeda tetapi masih dalam jangkauan expert system yang dibuat

Repository Dokumentasi

1. CLIPS

```
(deftemplate kotak-terbuka
   (slot location-x
      (type NUMBER)
      (default 0)
   )
   (slot location-y
      (type NUMBER)
      (default 0)
   )
   (slot contain
      (type NUMBER)
      (default -1)
   )
)

(deftemplate kotak-flag
   (slot id (default-dynamic (gensym*)))
   (slot location-x
      (type NUMBER)
      (default 0)
   )
   (slot location-y
      (type NUMBER)
      (default 0)
   )
)

(deftemplate kotak-tertutup
   (slot id (default-dynamic (gensym*)))
   (slot location-x
      (type NUMBER)
      (default 0)
   )
   (slot location-y
      (type NUMBER)
      (default 0)
   )
)
```

```
(deftemplate akan-buka-kotak
   (slot location-x
      (type NUMBER)
      (default 0)
   )
   (slot location-y
      (type NUMBER)
      (default 0)
   )
)

(deftemplate akan-flag
   (slot location-x
      (type NUMBER)
      (default 0)
   )
   (slot location-y
      (type NUMBER)
      (default 0)
   )
)

(deftemplate track-kotak
   (slot location-x
      (type NUMBER)
      (default 0)
   )
   (slot location-y
      (type NUMBER)
      (default 0)
   )
   (slot contain
      (type NUMBER)
      (default -1)
   )
   (multislot surrounding-flags)
   (multislot surrounding-unknown)
)

; Testing
;(deffacts init
;   (kotak-flag (location-x 0) (location-y 0))
;   (kotak-terbuka (location-x 0) (location-y 1) (contain 2))
;   (kotak-tertutup (location-x 0) (location-y 2))
;   (kotak-terbuka (location-x 1) (location-y 0) (contain 2))
;   (kotak-terbuka (location-x 1) (location-y 1) (contain 4))
;   (kotak-tertutup (location-x 1) (location-y 2))
;   (kotak-flag (location-x 2) (location-y 0))
;   (kotak-terbuka (location-x 2) (location-y 1) (contain 2))
;   (kotak-tertutup (location-x 2) (location-y 2))
;)
```

```
; --- Utils ---
; Flag all surrounding squares
(deffunction flag-surrounding (?x ?y)
   (do-for-all-facts ((?s kotak-tertutup)) (and (<= (abs (- ?s:location-x ?x)) 1) (<= (abs (- ?s:location-y ?y)) 1))
      (assert (akan-flag (location-x ?s:location-x) (location-y ?s:location-y)))
   )
)

; Reveal all surrounding squares
(deffunction reveal-surrounding (?x ?y)
   (do-for-all-facts ((?s kotak-tertutup)) (and (<= (abs (- ?s:location-x ?x)) 1) (<= (abs (- ?s:location-y ?y)) 1)) (assert (akan-buka-kotak (location-x ?s:location-x) (location-y ?s:location-y))))
)
```

```
(defrule akan-flag
   (declare (salience 30))
   ?a <- (akan-flag (location-x ?x) (location-y ?y))
   ?s <- (kotak-tertutup (location-x ?x) (location-y ?y))
   =>
   (retract ?s)
   (retract ?a)
   (assert (kotak-flag (location-x ?x) (location-y ?y)))
)

; --- Trackers ---
; Add flags to tracking list
(defrule add-flag
   (declare (salience 40))
   ?t <- (track-kotak (location-x ?x1) (location-y ?y1) (surrounding-flags $?s))
   (kotak-flag (id ?id) (location-x ?x2&:(<= (abs (- ?x2 ?x1)) 1)) (location-y ?y2&:(<= (abs (- ?y2 ?y1)) 1)))
   (test (and (not (member$ ?id ?s)) (not (and (= ?x1 ?x2) (= ?y1 ?y2)))))
   =>
   (modify ?t (surrounding-flags ?s ?id))
)

; Add unknown square to tracking list
(defrule add-unknown
   (declare (salience 40))
   ?t <- (track-kotak (location-x ?x1) (location-y ?y1) (surrounding-unknown $?s))
   (kotak-tertutup (id ?id) (location-x ?x2&:(<= (abs (- ?x2 ?x1)) 1)) (location-y ?y2&:(<= (abs (- ?y2 ?y1)) 1)))
   (test (and (not (member$ ?id ?s)) (not (and (= ?x1 ?x2) (= ?y1 ?y2)))))
   =>
   (modify ?t (surrounding-unknown ?s ?id))
)

; Remove newly known square from squares that tracks it
(defrule remove-unknown
   (declare (salience 40))
   ?t <- (track-kotak (location-x ?x1) (location-y ?y1) (surrounding-unknown $?b ?id $?a))
   (not (kotak-tertutup (id ?id) (location-x ?x2) (location-y ?y2)))
   =>
   (modify ?t (surrounding-unknown ?b ?a))
)

; Add tracker for each known squares
(defrule add-tracker
   (declare (salience 50))
   (kotak-terbuka (location-x ?x) (location-y ?y) (contain ?c))
   =>
   (assert (track-kotak (location-x ?x) (location-y ?y) (contain ?c)))
)
```

```
; --- Strategies ---
(defrule flags-eq-number
   (declare (salience 20))
   (kotak-terbuka (location-x ?x) (location-y ?y) (contain ?c))
   (track-kotak (location-x ?x) (location-y ?y) (surrounding-flags $?f))
   (test (= (length$ ?f) ?c))
   =>
   (reveal-surrounding ?x ?y)
)

(defrule squares-eq-number
   (declare (salience 20))
   (kotak-terbuka (location-x ?x) (location-y ?y) (contain ?c))
   (track-kotak (location-x ?x) (location-y ?y) (surrounding-flags $?f) (surrounding-unknown $?u))
   (test (= (+ (length$ ?f) (length$ ?u)) ?c))
   =>
   (flag-surrounding ?x ?y)
)


; --- 1-1 Pattern ---
; Edge on left
(defrule strategy11-horizontal-left
   (track-kotak (location-x ?x1) (location-y ?y) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x2&:(= ?x2 (+ ?x1 1))) (location-y ?y) (contain ?c2) (surrounding-flags $?f2))

   (not (exists (kotak-tertutup (location-x ?x3&:(= ?x3 (- ?x1 1))) (location-y ?y3&:(<= (abs (- ?y3 ?y)) 1)))))
   (test (and (= (- ?c1 (length$ ?f1)) 1) (= (- ?c2 (length$ ?f2)) 1)))
=>
   (bind ?loc (- ?x2 ?x1))
   (do-for-all-facts ((?s kotak-tertutup)) (and (= ?s:location-x (+ ?x2 1)) (<= (abs (- ?s:location-y ?y)) 1))
      (assert (akan-buka-kotak (location-x ?s:location-x) (location-y ?s:location-y)))
   )
)

; Edge on right
(defrule strategy11-horizontal-right
   (track-kotak (location-x ?x1) (location-y ?y) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x2&:(= ?x2 (- ?x1 1))) (location-y ?y) (contain ?c2) (surrounding-flags $?f2))

   (not (exists (kotak-tertutup (location-x ?x3&:(= ?x3 (+ ?x1 1))) (location-y ?y3&:(<= (abs (- ?y3 ?y)) 1)))))
   (test (and (= (- ?c1 (length$ ?f1)) 1) (= (- ?c2 (length$ ?f2)) 1)))
=>
   (bind ?loc (- ?x2 ?x1))
   (do-for-all-facts ((?s kotak-tertutup)) (and (= ?s:location-x (- ?x2 1)) (<= (abs (- ?s:location-y ?y)) 1))
      (printout t "Open: (" ?s:location-x "," ?s:location-y ")" crlf)
      (assert (akan-buka-kotak (location-x ?s:location-x) (location-y ?s:location-y)))
   )
)
```

```
; Edge above
(defrule strategy11-vertical-up
   (track-kotak (location-x ?x) (location-y ?y1) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x) (location-y ?y2&:(= ?y2 (+ ?y1 1))) (contain ?c2) (surrounding-flags $?f2))
   (not (exists (kotak-tertutup (location-x ?x3&:(<= (abs (- ?x3 ?x)) 1)) (location-y ?y3&:(= ?y3 (- ?y1 1))))))
   (test (and (= (- ?c1 (length$ ?f1)) 1) (= (- ?c2 (length$ ?f2)) 1)))
=>
   (bind ?loc (- ?y2 ?y1))
   (do-for-all-facts ((?s kotak-tertutup)) (and (= ?s:location-y (+ ?y2 1)) (<= (abs (- ?s:location-x ?x)) 1))
      (assert (akan-buka-kotak (location-x ?s:location-x) (location-y ?s:location-y)))
   )
)

; Edge below
(defrule strategy11-vertical-down
   (track-kotak (location-x ?x) (location-y ?y1) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x) (location-y ?y2&:(= ?y2 (- ?y1 1))) (contain ?c2) (surrounding-flags $?f2))
   (not (exists (kotak-tertutup (location-x ?x3&:(<= (abs (- ?x3 ?x)) 1)) (location-y ?y3&:(= ?y3 (+ ?y1 1))))))
   (test (and (= (- ?c1 (length$ ?f1)) 1) (= (- ?c2 (length$ ?f2)) 1)))
=>
   (bind ?loc (- ?y2 ?y1))
   (do-for-all-facts ((?s kotak-tertutup)) (and (= ?s:location-y (- ?y2 1)) (<= (abs (- ?s:location-x ?x)) 1))
      (printout t "Open: (" ?s:location-x "," ?s:location-y ")" crlf)
      (assert (akan-buka-kotak (location-x ?s:location-x) (location-y ?s:location-y)))
   )
)

; --- 1-2 Pattern ---
; 2 on the left
(defrule strategy12-horizontal-left
   (track-kotak (location-x ?x1) (location-y ?y) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x2&:(= ?x2 (+ ?x1 1))) (location-y ?y) (contain ?c2) (surrounding-flags $?f2))

   (kotak-tertutup (location-x ?x3&:(= ?x3 (- ?x1 1))) (location-y ?y3&:(<= (abs (- ?y3 ?y)) 1)))
   (not (exists (kotak-tertutup (location-x ?x3) (location-y ?y4&:(and (<> ?y3 ?y4) (<= (abs (- ?y4 ?y)) 1))))))

   (test (and (= (- ?c1 (length$ ?f1)) 2)
              (= (- ?c2 (length$ ?f2)) 1)))
   =>
   ;(printout t "1-2 hl" crlf)
   (assert (akan-flag (location-x ?x3) (location-y ?y3)))
)
```

```
; 2 on the right
(defrule strategy12-horizontal-right
   (track-kotak (location-x ?x1) (location-y ?y) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x2&:(= ?x2 (- ?x1 1))) (location-y ?y) (contain ?c2) (surrounding-flags $?f2))

   (kotak-tertutup (location-x ?x3&:(= ?x3 (+ ?x1 1))) (location-y ?y3&:(<= (abs (- ?y3 ?y)) 1)))
   (not (exists (kotak-tertutup (location-x ?x3) (location-y ?y4&:(and (<> ?y3 ?y4) (<= (abs (- ?y4 ?y)) 1))))))

   (test (and (= (- ?c1 (length$ ?f1)) 2)
              (= (- ?c2 (length$ ?f2)) 1)))
   =>
   ;(printout t "1-2 hr" crlf)
   (assert (akan-flag (location-x ?x3) (location-y ?y3)))
)

; 2 above
(defrule strategy12-vertical-up
   (track-kotak (location-x ?x) (location-y ?y1) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x) (location-y ?y2&:(= ?y2 (+ ?y1 1))) (contain ?c2) (surrounding-flags $?f2))

   (kotak-tertutup (location-x ?x3&:(<= (abs (- ?x3 ?x)) 1)) (location-y ?y3&:(= ?y3 (- ?y1 1))))
   (not (exists (kotak-tertutup (location-x ?x4&:(and (<> ?x3 ?x4) (<= (abs (- ?x4 ?x)) 1))) (location-y ?y3))))
   (test (and (= (- ?c1 (length$ ?f1)) 2)
              (= (- ?c2 (length$ ?f2)) 1)))
   =>
   (printout t "1-2 vu" crlf)
   (assert (akan-flag (location-x ?x3) (location-y ?y3)))
)

; 2 below
(defrule strategy12-vertical-down
   (track-kotak (location-x ?x) (location-y ?y1) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x) (location-y ?y2&:(= ?y2 (- ?y1 1))) (contain ?c2) (surrounding-flags $?f2))

   (kotak-tertutup (location-x ?x3&:(<= (abs (- ?x3 ?x)) 1)) (location-y ?y3&:(= ?y3 (+ ?y1 1))))
   (not (exists (kotak-tertutup (location-x ?x4&:(and (<> ?x3 ?x4) (<= (abs (- ?x4 ?x)) 1))) (location-y ?y3))))

   (test (and (= (- ?c1 (length$ ?f1)) 2)
              (= (- ?c2 (length$ ?f2)) 1)))
   =>
   ;(printout t "1-2 vd" crlf)
   (assert (akan-flag (location-x ?x3) (location-y ?y3)))
)
```

```
; --- 1-2-1 Pattern ---
; Horizontal 1-2-1
(defrule strategy121-horizontal
   (track-kotak (location-x ?x1) (location-y ?y) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x2&:(= (- ?x1 ?x2) 1)) (location-y ?y) (contain ?c2) (surrounding-flags $?f2))
   (track-kotak (location-x ?x3&:(= (- ?x1 ?x3) -1)) (location-y ?y) (contain ?c3) (surrounding-flags $?f3))
   (test (and (= (- ?c1 (length$ ?f1)) 2)
              (= (- ?c2 (length$ ?f2)) 1)
              (= (- ?c3 (length$ ?f3)) 1)))
   =>
   (do-for-all-facts ((?s kotak-tertutup)) (and (= ?s:location-x ?x1) (<= (abs (- ?s:location-y ?y)) 1)) (assert (akan-buka-kotak (location-x ?s:location-x) (location-y ?s:location-y))))
)

; Vertical 1-2-1
(defrule strategy121-vertical
   (track-kotak (location-x ?x) (location-y ?y1) (contain ?c1) (surrounding-flags $?f1))
   (track-kotak (location-x ?x) (location-y ?y2&:(= (- ?y1 ?y2) 1)) (contain ?c2) (surrounding-flags $?f2))
   (track-kotak (location-x ?x) (location-y ?y3&:(= (- ?y1 ?y3) -1)) (contain ?c3) (surrounding-flags $?f3))
   (test (and (= (- ?c1 (length$ ?f1)) 2)
              (= (- ?c2 (length$ ?f2)) 1)
              (= (- ?c3 (length$ ?f3)) 1)))
   =>
   (do-for-all-facts ((?s kotak-tertutup)) (and (<= (abs (- ?s:location-x ?x)) 1) (= ?s:location-y ?y1)) (assert (akan-buka-kotak (location-x ?s:location-x) (location-y ?s:location-y))))
)
```

2. Python

```python
#Dependency
import pygame
import random as acak
import sys
import clipsIO

sys.setrecursionlimit(1500)

def papanMinesweeper(size, bom, listBomX, listBomY):
    papan = [[0 for row in range(size)] for column in range(size)]

    for num in range(bom):
        x = listBomX[num]
        y = listBomY[num]
        papan[y][x] = 'X'

        if (x >=0 and x <= size-2) and (y >= 0 and y <= size-1):
            if papan[y][x+1] != 'X':
                papan[y][x+1] += 1 # center right
        if (x >=1 and x <= size-1) and (y >= 0 and y <= size-1):
            if papan[y][x-1] != 'X':
                papan[y][x-1] += 1 # center left
        if (x >= 1 and x <= size-1) and (y >= 1 and y <= size-1):
            if papan[y-1][x-1] != 'X':
                papan[y-1][x-1] += 1 # top left

        if (x >= 0 and x <= size-2) and (y >= 1 and y <= size-1):
            if papan[y-1][x+1] != 'X':
                papan[y-1][x+1] += 1 # top right
        if (x >= 0 and x <= size-1) and (y >= 1 and y <= size-1):
            if papan[y-1][x] != 'X':
                papan[y-1][x] += 1 # top center

        if (x >=0 and x <= size-2) and (y >= 0 and y <= size-2):
            if papan[y+1][x+1] != 'X':
                papan[y+1][x+1] += 1 # bottom right
        if (x >= 1 and x <= size-1) and (y >= 0 and y <= size-2):
            if papan[y+1][x-1] != 'X':
                papan[y+1][x-1] += 1 # bottom left
        if (x >= 0 and x <= size-1) and (y >= 0 and y <= size-2):
            if papan[y+1][x] != 'X':
                papan[y+1][x] += 1 # bottom center
    return papan

def papanUser(size):
    papanVisible = [['-' for row in range(size)] for column in range(size)]
    return papanVisible

def renderNonGui(papan):
    for row in papan:
        print(" ".join(str(cell) for cell in row) + "\n")
```

```
def bersihkanKosong(papanUser, papanAsli, x, y, size):
    if(papanAsli[y][x] == 0):
        if(papanUser[y][x] == '-'):
            papanUser[y][x] = papanAsli[y][x]
            #print("Masuk IF")
            if (x >=0 and x <= size-2) and (y >= 0 and y <= size-1):
                if papanAsli[y][x+1] == 0:
                    bersihkanKosong(papanUser, papanAsli, x+1, y, size) #tengah kanan
                else:
                    papanUser[y][x+1] = papanAsli[y][x+1]
            if (x >=1 and x <= size-1) and (y >= 0 and y <= size-1):
                if papanAsli[y][x-1] == 0:
                    bersihkanKosong(papanUser, papanAsli, x-1, y, size) #tengah kiri
                else:
                    papanUser[y][x-1] = papanAsli[y][x-1]
            if (x >= 1 and x <= size-1) and (y >= 1 and y <= size-1):
                if papanAsli[y-1][x-1] == 0:
                    bersihkanKosong(papanUser, papanAsli, x-1, y-1, size) #atas kiri
                else:
                    papanUser[y-1][x-1] = papanAsli[y-1][x-1]

            if (x >= 0 and x <= size-2) and (y >= 1 and y <= size-1):
                if papanAsli[y-1][x+1] == 0:
                    bersihkanKosong(papanUser, papanAsli, x+1, y-1, size) #atas kanan
                else:
                    papanUser[y-1][x+1] = papanAsli[y-1][x+1]
            if (x >= 0 and x <= size-1) and (y >= 1 and y <= size-1):
                if papanAsli[y-1][x] == 0:
                    bersihkanKosong(papanUser, papanAsli, x, y-1, size) #atas tengah
                else:
                    papanUser[y-1][x] = papanAsli[y-1][x]

            if (x >=0 and x <= size-2) and (y >= 0 and y <= size-2):
                if papanAsli[y+1][x+1] == 0:
                    bersihkanKosong(papanUser, papanAsli, x+1, y+1, size) #bawah kanan
                else:
                    papanUser[y+1][x+1] = papanAsli[y+1][x+1]
            if (x >= 1 and x <= size-1) and (y >= 0 and y <= size-2):
                if papanAsli[y+1][x-1] == 0:
                    bersihkanKosong(papanUser, papanAsli, x-1, y+1, size) #bawah kiri
                else:
                    papanUser[y+1][x-1] = papanAsli[y+1][x-1]
            if (x >= 0 and x <= size-1) and (y >= 0 and y <= size-2):
                if papanAsli[y+1][x] == 0:
                    bersihkanKosong(papanUser, papanAsli, x, y+1, size) #bawah tengah
                else:
                    papanUser[y+1][x] = papanAsli[y+1][x]
        papanAsli[y][x] = 0
        papanUser[y][x] = 0
    else:
        print("Masuk Else")
```

```
def isWon(papanAsli,size):
    papan = clipsIO.getKotakTerbuka(size)
    for row in range(size):
        for cell in range(size):
            isbom = papanAsli[row][cell]=='X'
            if (not papan[row][cell]): #Masih ada yang belum kebuka
                if (not isbom):
                    return False
            else:
                if (isbom):
                    return False
    return True

def gameStatus(score):
    print("Skor kamu adalah: " + str(score))
    isContinue = input("Coba lagi? (y/n) : ")
    if (isContinue == 'n'):
        return False
    return True

def main():
    #KONSTANTA
    arrBomX = []
    arrBomY = []

    #ALGORITMA
    f = open("input.txt","r")
    line = f.readline()
    ukuranPapan = int(line)
    print("Ukuran Papan = " + str(ukuranPapan))
    line = f.readline()
    jumlahBom = int(line)

    for i in range(jumlahBom):
        line = f.readline()
        bomX, bomY = tuple(map(int,line.split(',')))
        arrBomX.append(bomX)
        arrBomY.append(bomY)

    GameStatus = True
    while GameStatus:
        papanReal = papanMinesweeper(ukuranPapan, jumlahBom, arrBomX, arrBomY)
        papanVisible = papanUser(ukuranPapan)
        for i in range(ukuranPapan):
            for j in range(ukuranPapan):
                if (papanReal[j][i] == 'X'):
                    clipsIO.addKotak(i, j, -1)
                else:
                    clipsIO.addKotak(i, j, int(papanReal[j][i]))
        skor = 0
        renderNonGui(papanReal)
```

```
while True:
    if isWon(papanVisible, ukuranPapan) == False:
        print("Mau buka kotak yang mana?")
        x = int(input("X : "))
        y = int(input("Y : "))

        if (papanReal[y][x] == 'X'):
            print("DUARRRR!!!")
            renderNonGui(papanReal)
            GameStatus = gameStatus(skor)
            break
        else:
            if (papanReal[y][x] == 0):
                bersihkanKosong(papanVisible, papanReal, x, y, ukuranPapan)
                renderNonGui(papanVisible)
                skor += 1
            else:
                papanVisible[y][x] = papanReal[y][x]
                renderNonGui(papanVisible)
                skor += 1
    else:
        renderNonGui(papanVisible)
        print("Hore menang!!")
        GameStatus = gameStatus(skor)
        break
```

User Manual

Langkah untuk menjalankan program minesweeper:

1. Diperlukan untuk menginstall python dan library clipspy. Install library clipspy dilakukan melalui perintah berikut:

   Pip install clipspy

2. Input ukuran board, jumlah bom, beserta koordinat dilakukan di input.txt
3. Buka folder minesweeper dan buka file main.py pada VS code
4. Kemudian run file main.py pada VS code

Proses Updating dan Inferencing atas Fakta yang Terlibat

Proses updating dan inferencing atas fakta yang terlibat pada Minesweeper:

a. Generate rule dan fact pada minesweeper yang dilakukan miner.clp
b. CLIPS akan memainkan permainan minesweeper dengan menebak posisi bom setelah klik pertama pada koordinat (0,0)
c. CLIPS akan melihat kotak yang terbuka

d. CLIPS akan menentukan kotak yang akan dibuka dan kotak yang di flag
e. Ketika CLIPS bertemu dengan pattern 1-1 yang dimulai dari sisi, maka kotak ketiga pasti kosong
f. Ketika CLIPS bertemu dengan pattern 1-2, maka kotak ketiga pasti bom
g. Ketika CLIPS bertemu dengan pattern yang lain, pattern yang lain tersebut merupakan kombinasi dari 1-1 atau 1-2 atau kombinasi keduanya
h. CLIPS akan mengulangi tahapan hingga menang atau kalah karena memencet bom