Proyecto Final de Robots Móviles
Planeación de Acciones Usando un Sistema Basado en Reglas

Objetivo: Familiarizar al alumno con la planeación de acciones usando sistemas basados en reglas.

Desarrollo: Para cada uno de los siguientes apartados, realizar los programas que se piden.

Duración: cuatro semanas

1.- Descargue de la página del GitHub la versión de CLIPS con sockets TCP: https://github.com/kyordhel/tcpclips60

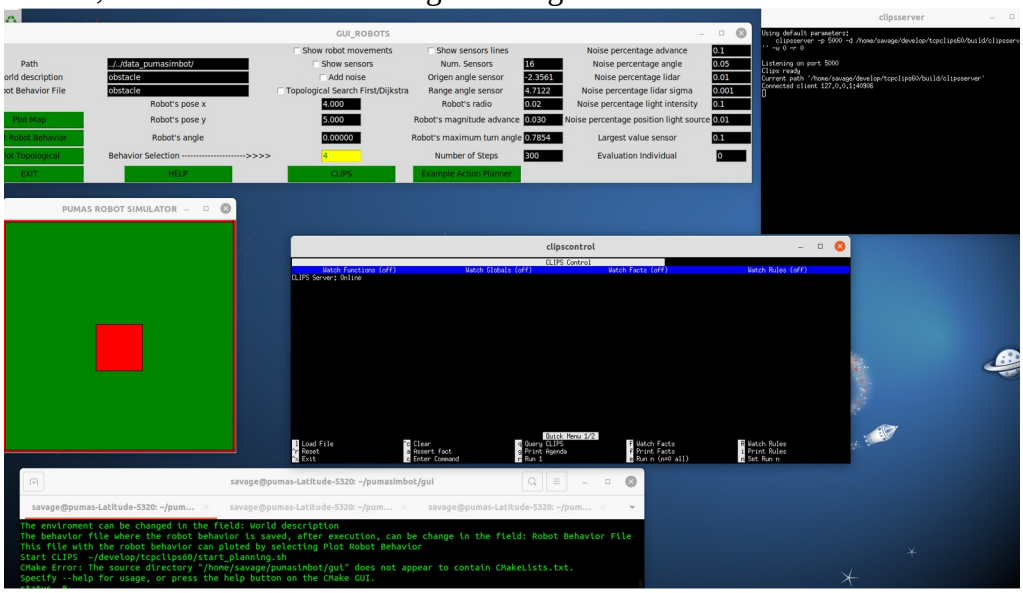
Paso 0: Instalación de dependencias, en una terminal, ejecutar:
apt install build-essential cmake libboost-all-dev libncurses-dev

Paso 1: Clonar el repositorio, en una terminal, estando en el directorio home:
mkdir -p ~/develop
cd develop
git clone https://github.com/kyordhel/tcpclips60.git

Paso 2: Construir el ejecutable:
cd tcpclips60/build
cmake .. && make

2.- Cargue la nueva versión del simulador de robots pumassimbot.zip y data_pumasimbot.zip como se indicó en la práctica número 1 del curso.

3.- Una vez instalado la nueva versión del simulador y la versión de Clips con sockets seleccione el botón de CLIPS en el GUI del simulador. Deberán aparecer dos terminales nuevas, cómo se muestra en la siguiente figura:



En la terminal, clipscontrol, es donde se le dan los comandos a Clips y en la otra terminal es clipsserver que es donde se ve la ejecución del código de Clips.
Para hacer las letras más grandes en cada una de las terminales presione la tecla ctrl y el botón derecho del mouse y seleccione la opción Huge.
El apéndice A contiene el código de un sistema que controla la manipulación de cubos en diferentes configuraciones usando el lenguaje basado en reglas CLIPS con una representación del medio ambiente con pilas.
Edite este código en el archivo cubes_stacks.clp y colóquelo en el directorio:
/home/usuario/pumasimbot/action_planner/ViRBot_Planning_Sockets/
en donde usuario es la raiz del usuario en su computadora. Para probar este código en la interface de Clips primero indique el lugar de donde se cargaran los archivos de Clips, en la ventana de clipscontrol con la opción "p", (Set clp path) indique el path:
/home/usuario/pumasimbot/action_planner/ViRBot_Planning_Sockets/

Cargue este código con la opción "l" Load File en la ventana clipscontrol. Para ver los hechos que se van creando, usar la opción "F", las reglas que se van ejecutando con "R". Dar reset a Clips con ctrl r y se ejecuta paso a paso con la tecla r.

Entienda el funcionamiento del código en cubes_stacks.clp y modifiquelo para que resuelva el problema con tres pilas:

(stack A B C)
(stack D E F)
(stack G H I)

Y con el siguiente objetivo:

(goal A D G B E H C F I)

Dejando como resultado:

(stack-final A D G B E H C F I)

4.- Seleccionando el botón "Example Action Planner" muestra un ejemplo de planeación de acciones usando código de Clips para generar la planeación de acciones.
También se puede ejecutar esta planeación de acciones con la opción de comportamiento 9 en el GUI del simulador, con el medio ambiente "final" y colocando el número de pasos en 15000 si desea ver todo el proceso completo. Para empezar a ver la ejecución escoger con el botón izquierdo del mouse la posición del robot y con el derecho un destino. Si se desea ver solamente cómo Clips genera el plan coloque el número de pasos en 1 y después seleccione en la terminal de clipscontrol para observar los hechos y las reglas, de está forma se puede observar cómo el ejecutor de acciones le va indicando las operaciones que se van haciendo, para así modificar la base de conocimiento del sistema.

El código de Clips en donde se genera el plan está en:
~/pumasimbot/action_planner/ViRBot_Planning_Sockets/

El archivo virbot_initial_state.clp contiene la información de los objetos, cuartos, muebles, robots, etc.
Los objetivos de cómo se van a ir colocando los objetos está determinado por los hechos:

(goal-stack 5 deposit desk book)
(goal-stack 4 studio desk hammer)
(goal-stack 3 bedroom bed soap apple)
(goal-stack 2 service service_table perfume shampoo)
(goal-stack 1 kitchen cold_storage sushi milk)

Una vez que el plan es generado, con el código de Clips en los archivos *.clp que se encuentran en el directorio ~/pumasimbot/action_planner/ViRBot_Planning_Sockets/, el plan se envía por sockets al executor the planes, cuyo código se encuentra en el directorio:
~/pumasimbot/action_planner/

Por ejemplo, para llevar la leche que se encuentra en el corredor en el área de deposito a la cocina y al refrigerador, se genera el siguiente plan:

1. goto corridor deposit.
El robot debe de ir del lugar en donde se encuentra al cuarto llamado corridor y a la zona deposit. Aquí se usa el planeador de movimientos usando comportamientos para evadir obstáculos desconocidos y el algoritmo de Dijkstra.

2. find-object milk.
Con el sistema de visión simulado se encuentra el objeto leche.

3. mv milk.
El robot se acerca a la leche.

4. grab milk.
El robot toma la leche con su manipulador simulado.

5. goto kitchen cold_storage.
El robot se dirije a la cocina en donde está el refrigerador.

6. find-object freespace.
El robot encuentra un espacio libre en el refrigerador en donde poner la leche.

7. go freespace.
Se acerca al espacio libre en donde pondrá la leche.

8. drop milk.
Suelta la leche.

Entienda en forma general como está funcionando este sistema de planeación y ejecución.

5.- En el apéndice B, figura 1, se muestra el medio ambiente con los nombres de los cuartos en donde opera el robot. En la figura 2 en el cuarto “Studio” se muestran la posición de los objetos A, B y C, en el cuarto denominado “Corridor” se encuentran los objetos D, E y F. Estos objetos serán relocalizados de acuerdo a la figura 3. Para poder mover los objetos y colocarlos en otro lugar éstos deberán ser movidos siguiendo un orden, es decir que para mover al objeto C se necesita mover primero los objetos A y B

y colocarlos en su posición final también se necesita hacerlo siguiendo un orden preestablecido.

Modifique las posiciones de los objetos en los archivos virbot_initial_state.clp e virbot_initial_state.txt que se encuentran en el directorio:
~/pumasimbot/action_planner/ ViRBot_Planning_Sockets.

Prueba su sistema con el simulador mostrando la planeación de acciones.

APÉNDICE A
CLIPS Mundo de los Bloques

```
;;;=========================================================
;;;   Programa del Mundo de los Bloques usando pilas
;;;   cubes_stacks.clp
;;;=========================================================

(deftemplate goal (slot move)(slot on-top-of))

(deffacts initial-state
      (get-initial-state stacks 2)
      (stack A B C)
      (stack D E F)
      (goal (move C)(on-top-of F))
)

(defrule move-directly
      ?goal <- (goal (move ?block1) (on-top-of ?block2))
      ?stack-1 <- (stack ?block1 $?rest1)
      ?stack-2 <- (stack ?block2 $?rest2)
      =>
      (retract ?goal ?stack-1 ?stack-2)
      (assert (stack $?rest1))
      (assert (stack ?block1 ?block2 $?rest2))
      (printout t ?block1 " moved on top of " ?block2 "." crlf)
)

(defrule move-to-floor
      ?goal <- (goal (move ?block1) (on-top-of floor))
      ?stack-1 <- (stack ?block1 $?rest)
      =>
      (retract ?goal ?stack-1)
      (assert (stack ?block1))
      (assert (stack $?rest))

      (printout t ?block1 " moved on top of floor. " crlf)
)


(defrule clear-upper-block
      (goal (move ?block1))
      (stack ?top  $? ?block1 $?)
=>
      (assert (goal (move ?top)(on-top-of floor)))
)
```

```
(defrule clear-lower-block
      (goal (on-top-of ?block1))
      (stack ?top  $? ?block1 $?)
      =>
      (assert (goal (move ?top)(on-top-of floor)))
)
```

APENDICE B
Medio ambiente en donde opera el robot

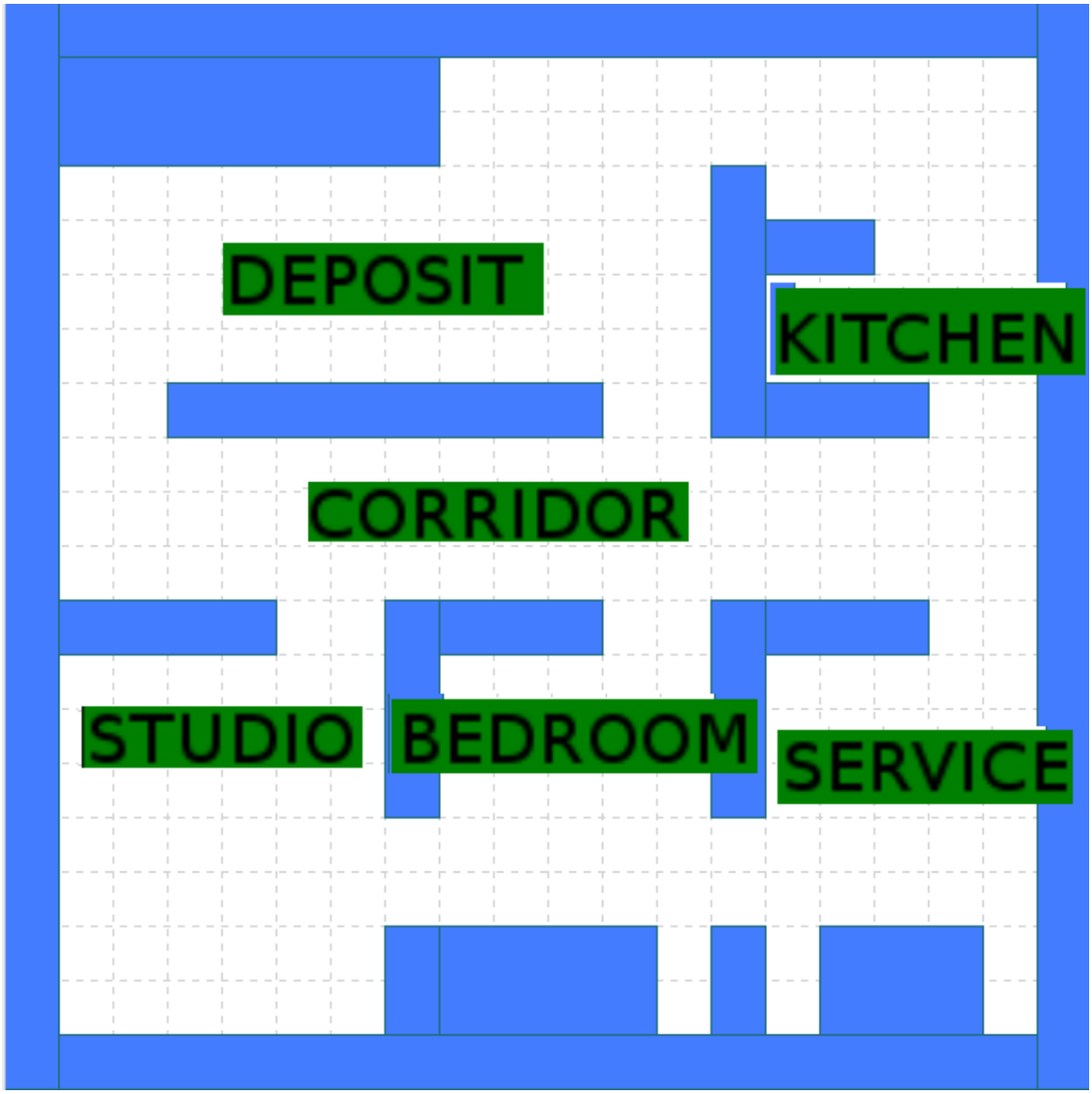


Figura 1.

Configuración original y final de los cubos

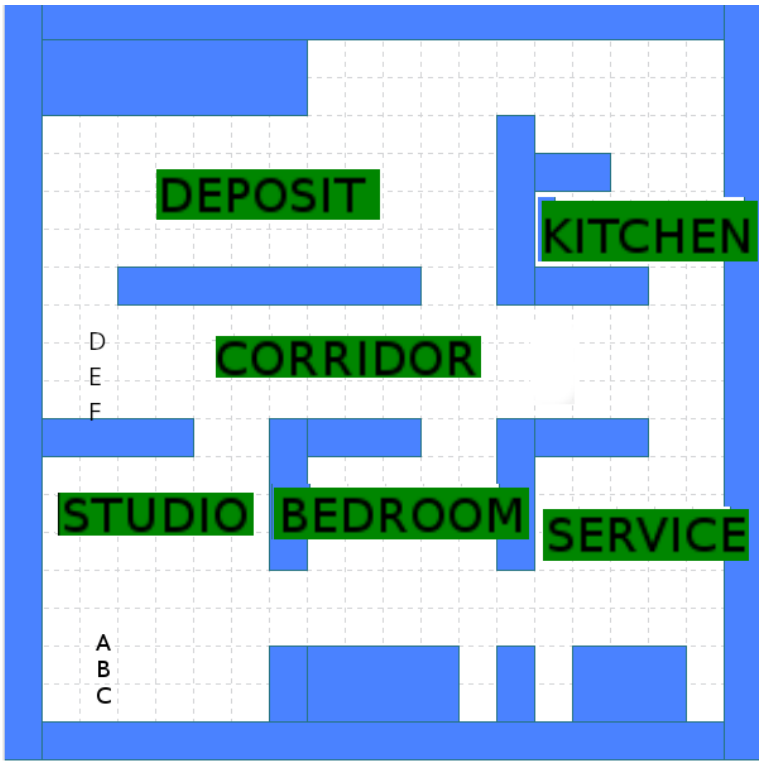


Figura 2.

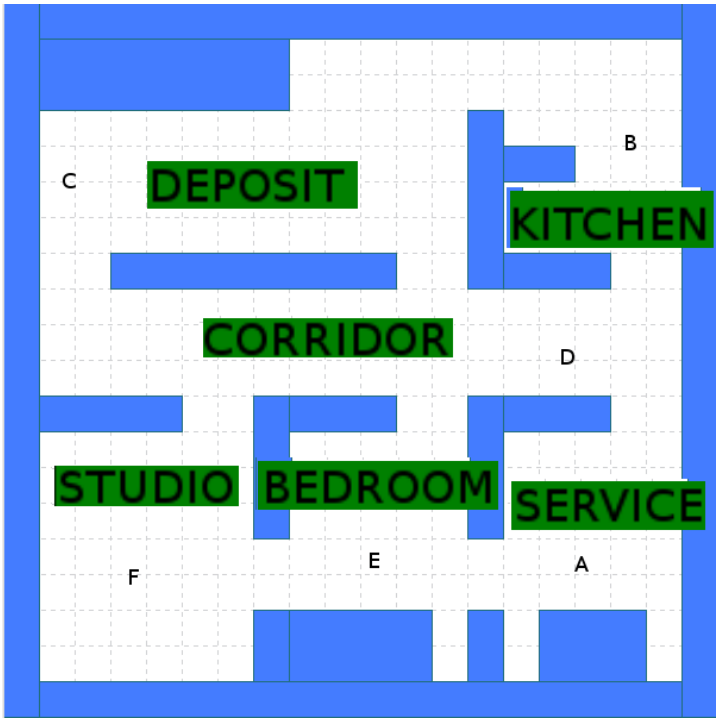


Figura 3.