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: Tres semanas

1.- Descargue de la página del curso el archivo proyecto_final_2024.zip y descomprima este archivo. En el se encuentra el archivo ejecutable del sistema clips, es posible que se necesiten modificar los permisos de ejecución de este archivo, eso se hace con el comando: chmod 777 clips.
El archivo cubes_operators.dat contiene los comandos para cargar los siguientes archivos de clips: objects_deftemplates.clp, initial_state_final.clp y cubes_operators.clp

Pruebe en una terminal el funcionamiento de CLIPS, ejecutando el comando:
./clips -f cubes_operators.dat

Para poder hacer "debug" de un código hecho en el lenguaje de CLIPS, en el "prompt" de clips ponga los siguientes comandos uno a la vez:
(watch rules)
(watch facts)
(reset).
Después ejecute el código paso a paso con (run 1).

2.- 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.
Este código también se encuentra en el archivo cubes_operators.clp. Pruebe este código y entienda su funcionamiento.

3.- 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 cubos A, B y C, en el cuarto denominado "Corridor" se encuentran los cubes 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 bloque C se necesita mover primero los bloques A y B y colocarlos en su posición final también se necesita hacerlo siguiendo un orden preestablecido.

Configure un sistema de movimiento de objetos de un lugar a otro usando CLIPS, teniendo reglas (axiomas), operadores y una representación del mundo con hechos.

Este sistema deberá ser flexible con respecto a solucionar cualquier configuración inicial y final, la salida de éste será un archivo utilizando los siguientes operadores:

goto room
find object
grasp object
release object

Se pueden conseguir las posiciones de los cuartos que se encuentran en el archivo initial_state_final.clp cuando se da reset y aparecen como hechos.

4.- Pruebe su sistema en el simulador utilizado durante el curso, leyendo el archivo generado en el punto anterior y ejecutando los comandos. Para la planeación de movimientos del robot entre cuartos use una red topológica y algoritmos de búsqueda, tales como el algoritmo de Dijkstra, A*, etc. Para obstáculos desconocidos utilice cualquiera de las siguientes técnicas: algoritmos de máquinas de estados o campos potenciales. Asuma que los operandos find, grasp y release los ejecuta el robot correctamente en el simulador.

APÉNDICE A
CLIPS Mundo de los Bloques

```
;;;=========================================================
;;;   Programa del Mundo de los Bloques
;;;   Para ejecutarlo, solamente carguelo en CLIPS:
;;;      clips -f cubes_operators.clp
;;;   de (reset) y ejecutelo con (run)
;;;   Los commandos que se le daran al robot quedan en el archivo
commnands.dat
;;;=========================================================

(deftemplate on-top-of
      (slot upper)
      (slot lower)
)

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

(deffacts initial-state
      (block A)
      (block B)
      (block C)
      (block D)
      (block E)
      (block F)
      (on-top-of (upper nothing)(lower A))
      (on-top-of (upper A)(lower B))
      (on-top-of (upper B)(lower C))
      (on-top-of (upper C)(lower floor))
      (on-top-of (upper nothing)(lower D))
      (on-top-of (upper D)(lower E))
      (on-top-of (upper E)(lower F))
      (on-top-of (upper F)(lower floor))
      ;(goal (move C)(on-top-of E))
      (goal (move F)(on-top-of C))
)
```

```
; it opens the file to save the robot commands
(defrule open-file
	(declare (salience 100))
	?f <- (initial-fact)

	=>
	(retract ?f)
	(open commands.dat output-file "w")
	(assert (file-commands output-file))
	(assert (file-name commands.dat))
)


(defrule move-directly
	?goal <- (goal (move ?block1) (on-top-of ?block2))
	(block ?block1)
	(block ?block2)
	(on-top-of (upper nothing) (lower ?block1))
	?stack-1 <- (on-top-of (upper ?block1)(lower ?block3))
	?stack-2 <- (on-top-of (upper nothing)(lower ?block2))
	(file-commands ?file)
	=>
	(retract ?goal ?stack-1 ?stack-2)
	(assert (on-top-of (upper ?block1)(lower ?block2))
		(on-top-of (upper nothing)(lower ?block3)))
	(printout t ?block1 " moved on top of " ?block2 "." crlf)
	(printout ?file "OPERATOR move " ?block1 " " ?block2  crlf)
)


(defrule move-to-floor
	?goal <- (goal (move ?block1) (on-top-of floor))
	(block ?block1)
	(on-top-of (upper nothing) (lower ?block1))
	?stack <- (on-top-of (upper ?block1) (lower ?block2))
	(file-commands ?file)
	=>
	(retract ?goal ?stack)
	(assert (on-top-of (upper ?block1)(lower floor))
		(on-top-of (upper nothing)(lower ?block2)))
	(printout t ?block1 " moved on top of the floor. " crlf)
	(printout ?file "OPERATOR move " ?block1 " floor" crlf)
```

```
)


(defrule clear-upper-block
	(goal (move ?block1))
	(block ?block1)
	(on-top-of (upper ?block2) (lower ?block1))
	(block ?block2)
	=>
	(assert (goal (move ?block2)(on-top-of floor)))
)


(defrule clear-lower-block
	(goal (on-top-of ?block1))
	(block ?block1)
	(on-top-of (upper ?block2) (lower ?block1))
	(block ?block2)
	=>
	(assert (goal (move ?block2)(on-top-of floor)))
)

(defrule finish-planning
	(declare (salience -400))
	?f <- (file-commands ?file)
	(file-name ?name)
	=>
	(retract ?f)
	(printout t crlf "finish planner" crlf)
	(printout t "Commands in file " ?name crlf)
	(printout ?file "finish planner" crlf)
	(close ?file)

)
```

APENDICE B
Medio ambiente en donde opera el robot

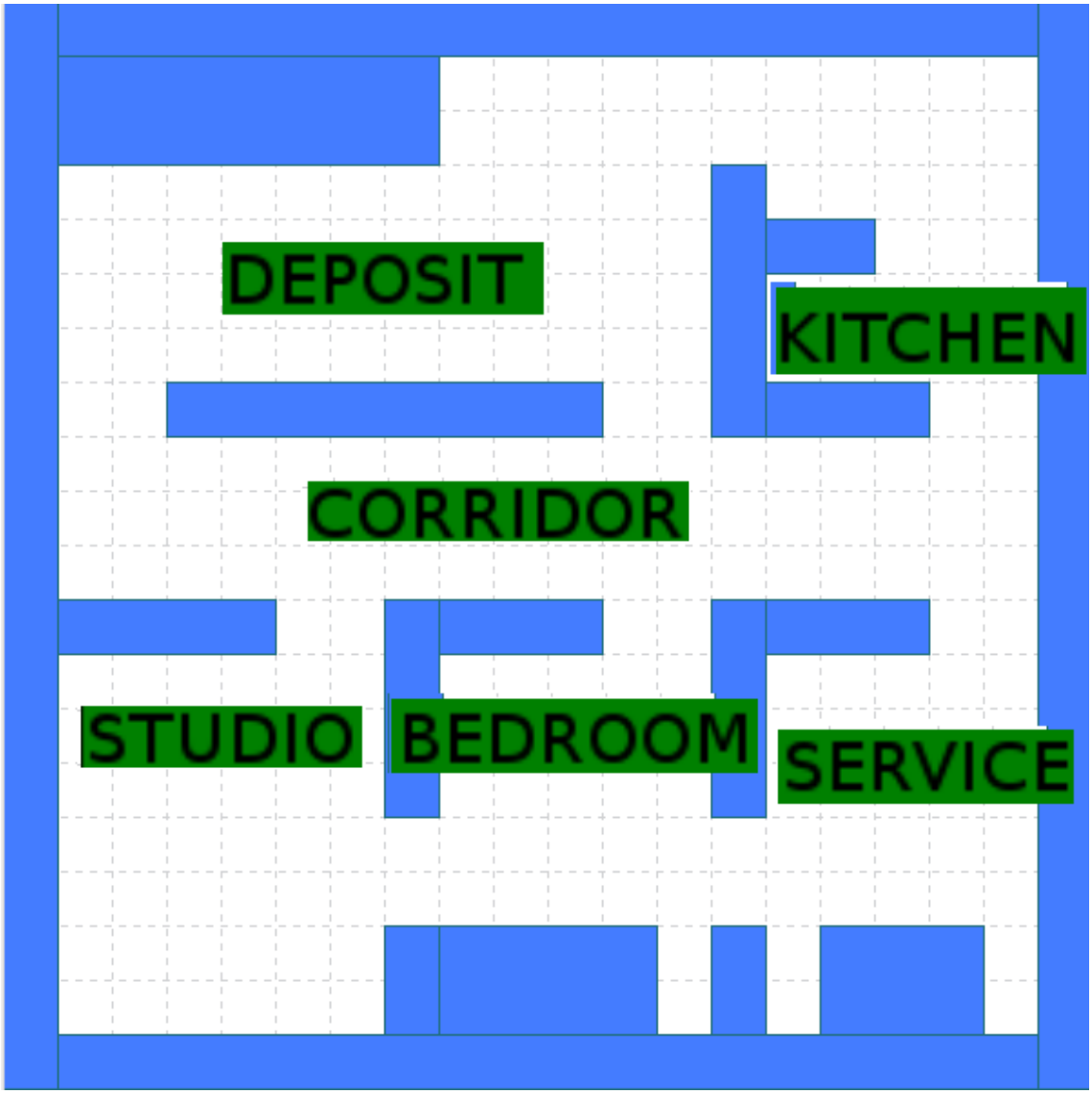

Figura 1.

Configuración original y final de los cubos

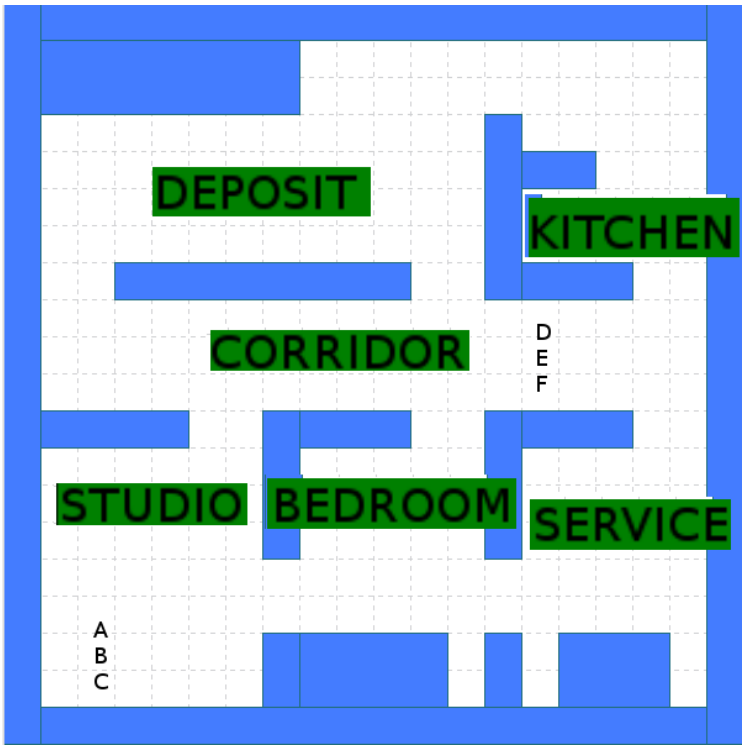

Figura 2.

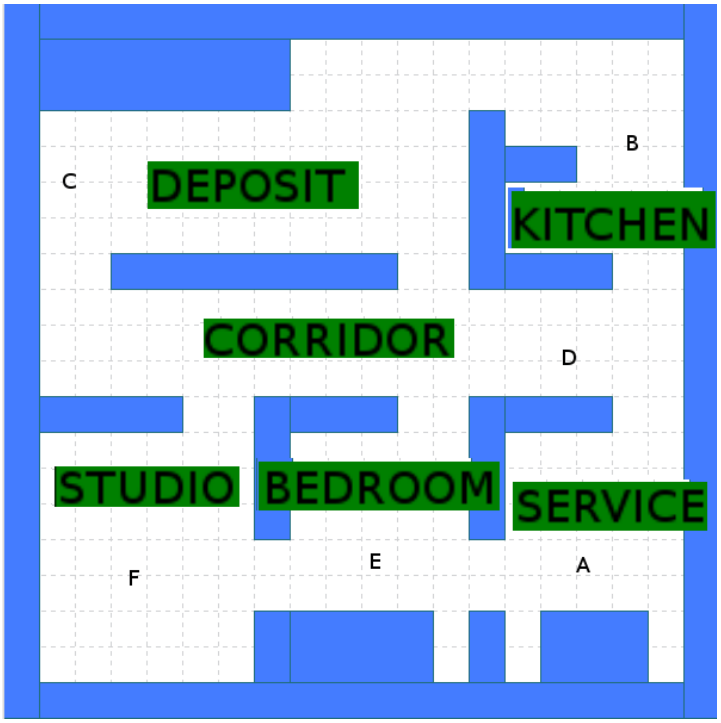

Figura 3.