Laboratorio de Casas Inteligentes
Uso de Técnicas de Inteligencia Artificial para Operar una Casa
Practica No. 8

Objetivo: Usar un sistema basado en reglas (CLIPS) para instrumentar técnicas de inteligencia artificial en la operación de una casa.

Desarrollo: Para cada uno de los siguientes apartados, realizar el software que se pide.
Duración: Dos semanas

1. CLIPS es una herramienta de que permite desarrollar técnicas de inteligencia artificial utilizando el concepto de sistemas expertos. La versión de Clips que se utilizará es la que combina Python con Clips: pyCLIPS. Para instalar pyCLIPS entrar a: http://pyclips.sourceforge.net/web/ y ahí buscar el link que dice "Downloadable files". Esto los lleva a un sitio de sourceforge donde está el repositorio de pyCLIPS. Entrar en la carpeta de **debian_packages**, bajar la que corresponda a su arquitectura (32 ó 64 bits) y cuando lo hayan bajado, ejecutar el siguiente comando:

```
dpkg -i package-name-here.deb
```

Probablemente necesiten permisos de superusuario.
Para ejecutarlo como superusuario pueden hacer 3 cosas:
- Ejecutarlo con sudo, si son sudoers.
- Ejecutar su -c "<comando>"
- Cambiarse a sesión de root con el comando su - y después ejecutarlo.

Si la instalación automática no funciona, se puede hacer en forma manual:

Bajar el .tar.gz que contiene el código fuente de la carpeta que dice **pyclips**.

1. Descomprímalo en alguna carpeta (les recomiendo hacer una carpeta pyclips_src y ahí meterlo) con el comando:
   ```
   tar -xzvf nombre-comprimido.tar.gz
   ```
2. Identifiquen y borren una carpeta llamada **clipssrc** o algo parecido. (El script de instalación descarga el código fuente si no lo encuentra en la misma carpeta)
3. Ejecuten los siguientes comandos:
   (Probablemente necesitarán permisos de superusuario para el segundo)
   ```
   python setup.py build

   python setup.py install
   ```

2. Descargue en la página de las prácticas el archivo denominado Example_Clips, el cual contiene ejemplos en CLIPS los cuales se conectan con el Blackboard utilizando variables compartidas.
Coloque este archivo en el directorio ~/Blackboard creado en las practicas anteriores.

Descomprimalo usando los siguientes commandos:
$ gunzip example_clips.tar.gz
$ tar xvf example_clips.tar

Estos comandos deberan haber creado el directorio example_clips

3. Pruebe que pyCLIPS quede integrado en la estructura del Blackboard:

* Open an xterm and change to the following directory
$ cd ~/Blackboard/example_clips

* Execute Blackboard.exe with mono
$ mono ~/Blackboard/BlackBoard_bin/Blackboard.exe BB_CLIPS.xml

* Open another xterm in the same directory and execute the python application
$ python test_python/test_python.py

* Open another xterm in the same directory and execute the python-clips application
$ python BBCLIPS/BBCLIPS.py

In the menu load clips_src/test_clips.clp
Select reset button and run

* Open another xterm and execute the Arduino application
$ ~/Blackboard/example_arduino_C++/bin/sensor_actuator

Observe que los datos obtenidos por esta aplicación, leídos por el puerto serial y que son generados por el Arduino, son enviados a través del Blackboard al módulo de CLIPS, este módulo procesa la información y dependiendo de los valores envía al módulo creado por test_python.py comandos para los actuadores. En el Apendice A se encuentra el código de clips test_clips.clp, entienda este código.

4. Modifique el código de test_clips.clp para que se efectúen varias acciones en los actuadores.
Por ejemplo si la temperatura es más grande que cierto umbral prender un motor; si la intensidad de luz esta abajo de cierto umbral se prenda una lampara, etc.

APENDICE A

```
;*****************************************************
;*                                                   *
;*      test_clips.clp                               *
;*                                                   *
;*                  J.Savage, UNAM                   *
;*                  8/11/14                          *
;*                                                   *
;*****************************************************

;*****************************************************
;*      Deftemplates definitions                     *
;*****************************************************
(deftemplate device
      (field name
            (type SYMBOL)
            (default nil)
      )
      (field category
           (type SYMBOL)
      )
      (multifield type
            (type SYMBOL)
      )
      (field temp
            (type NUMBER)
            (default 0)
      )
      (field magnet
            (type NUMBER)
            (default 0)
      )
      (multifield acelerometer
            (type NUMBER)
            (default 0)
      )
      (field photo
            (type NUMBER)
            (default 0)
      )
      (field motor
            (type NUMBER)
            (default 0)
      )
      (field light
            (type NUMBER)
            (default 0)
      )
      (field airconditioner
            (type NUMBER)
            (default 0)
      )
      (field movement
            (type NUMBER)
            (default 0)
      )
```

```
)


(deftemplate room
	(field name
		(type SYMBOL)
		(default nil)
	)
	(multifield devices
		(type SYMBOL)
	)
	(field type
		(type SYMBOL)
	)
)

;*************************************************
;*	Initial Facts                                *
;*************************************************
(deffacts Initial-state

; Room's definitions
	(room (name bedroom)(type dormitory )(devices A1 A2 A3 A4))
	(room (name livingroom)(type dormitory )(devices B1 B2 B3 B4))

; Device's definitions
	(device (name A1)(category sensor)(type temp magnet acelerometer photo ))
	(device (name A2)(category actuator)(type motor ))
	(device (name A3)(category sensor)(type movement ))
	(device (name A4)(category actuator)(type airconditioner ))

	(device (name B1)(category sensor)(type temp magnet acelerometer photo ))
	(device (name B2)(category actuator)(type light ))
	(device (name B3)(category sensor)(type movement ))
	(device (name B4)(category actuator)(type motor ))
	(setting temp 1.59)
)


;*************************************************
;*	BLACKBOARD  RULES                            *
;*************************************************

; This rule subscribes to the shared variables after reset
(defrule subscribe-variables
	?f <-(initial-fact)
	=>
	(retract ?f)

	(create-shared_var "string" "sensor")
	(write-shared_var "string" "sensor" "Updating var from CLIPS")
	(subscribe_to-shared_var "sensor" )

	(create-shared_var "string" "actuator")
	(write-shared_var "string" "actuator" "Updating Actuator from CLIPS")
	(subscribe_to-shared_var "actuator" )

	(assert (num 1))
```

```
)


; This rule is activated when a shared variable is modified
(defrule shared-variable
	(declare (salience 100))
	?f <- (BB_sv_updated ?shared_var ?value)
	?f1 <- (num ?num)
	=>
	(retract ?f ?f1)
	(bind ?var (explode$ ?shared_var))
	(bind $?prms (explode$ ?value))
	(assert (received ?var $?prms))
	(assert (num (+ ?num 1)) )
	;(printout t "shared variable " ?var " value " $?prms " " ?num crlf)
	;(printout t "num  " ?num crlf)
)

(defrule shared-variable-actuator
	?f <- (received actuator $?prms)
	=>
	(retract ?f)
	(assert (actuator $?prms))
	;(printout t "ACTUATOR " $?prms crlf)
)


(defrule shared-variable-sensor
	?f <- (received sensor $?prms)
	=>
	(retract ?f)
	(assert (sensor $?prms))
	;(printout t "SENSOR " $?prms crlf)
)



;*************************************************
;*		HOUSE  RULES                             *
;*************************************************

(defrule temperature_on
	?f <- (sensor ?device temp ?value $?date )
	?f1 <- (device (name ?device) (category sensor) (type $?before temp $?after) (temp ?))
	?f2 <- (device (name ?device1) (category actuator) (type airconditioner) (airconditioner 0))
	(room (name ?room)(devices $? ?device $?))
	(room (name ?room1)(devices $? ?device1 $?))
	(setting temp ?set_temp)
	(num ?num)
	=>
	(retract ?f)
	(modify ?f1 (temp ?value))

	(bind ?value_str (str-cat "The temperature in device " ?device " located in room " ?room " is equal
				to " ?value " (" ?num ")" ))
	(printout t ?value_str crlf)

	(if (> ?value ?set_temp) then
```

```
        (modify ?f2 (airconditioner 1))
        (bind ?param (str-cat ?device1 " turn on"))
        (write-shared_var "string" "actuator" ?param)
        (printout t ?param crlf)
    )
)


(defrule temperature_off
    ?f <- (sensor ?device temp ?value $?date )
    ?f1 <- (device (name ?device) (category sensor) (type $?before temp $?after) (temp ?))
    ?f2 <- (device (name ?device1) (category actuator) (type airconditioner) (airconditioner 1))
    (room (name ?room)(devices $? ?device $?))
    (room (name ?room1)(devices $? ?device1 $?))
    (setting temp ?set_temp)
    (num ?num)
    =>
    (retract ?f)
    (modify ?f1 (temp ?value))

    (bind ?value_str (str-cat "The temperature in device " ?device " located in room " ?room " is equal
                        to " ?value " (" ?num ")" ))
    (printout t ?value_str crlf)

    (if (< ?value ?set_temp) then
        (modify ?f2 (airconditioner 0))
        (bind ?param (str-cat ?device1 " turn off"))
        (write-shared_var "string" "actuator" ?param)
        (printout t ?param crlf)
    )
)


(defrule magnet_0
    ?f <- (sensor ?device magnet 0 $?date )
    ?f1 <- (device (name ?device) (category sensor) (type $?before magnet $?after) (magnet ?))
    ?f2 <- (device (name ?device1) (category actuator) (type light) (light 0))
    (room (name ?room)(devices $? ?device $?))
    (room (name ?room1)(devices $? ?device1 $?))
    (num ?num)
    =>
    (retract ?f)
    (modify ?f1 (magnet 0))
    (modify ?f2 (light 1))
    (bind ?value_str (str-cat "The magnet sensor in device " ?device " located in room " ?room " is
                                                                    equal to 0 (" ?num ")" ))
    (printout t ?value_str crlf)

    (bind ?param (str-cat ?device1 " turn on"))
    (write-shared_var "string" "actuator" ?param)
    (printout t ?param crlf)
)


(defrule magnet_1
    ?f <- (sensor ?device magnet 1 $?date )
    ?f1 <- (device (name ?device) (category sensor) (type $?before magnet $?after) (magnet ?))
    ?f2 <- (device (name ?device1) (category actuator) (type light) (light 1))
```

```
	(room (name ?room)(devices $? ?device $?))
	(room (name ?room1)(devices $? ?device1 $?))
	(num ?num)
	=>
	(retract ?f)
	(modify ?f1 (magnet 1))
	(modify ?f2 (light 0))
	(bind ?value_str (str-cat "The magnet sensor in device " ?device " located in room " ?room " is
                                                                    equal to 1 (" ?num ")" ))
	(printout t ?value_str crlf)

	(bind ?param (str-cat ?device1 " turn off"))
	(write-shared_var "string" "actuator" ?param)
	(printout t ?param crlf)
)


(defrule acelerometer
	?f <- (sensor ?device acelerometer ?valuex ?valuey ?valuez $?date )
	?f1 <- (device (name ?device) (category sensor) (type $?before acelerometer $?after)
                                                                    (acelerometer $?))
	(room (name ?room)(devices $? ?device $?))
	(num ?num)
	=>
	(retract ?f)
	(modify ?f1 (acelerometer ?valuex ?valuey ?valuez))
	(bind ?value_str (str-cat "The acelerometer in device " ?device " located in room " ?room " is
	    equal to " ?valuex " " ?valuey " " ?valuez " (" ?num ")" ))
	(printout t ?value_str crlf)

)
```