Memoria TIA
Práctica 1: Jess

Ramon Ruiz Dolz
4CO21

Dominio de aplicación

En este primer apartado se definirá el dominio de aplicación del sistema experto desarrollado en Jess para la primera sesión de prácticas de laboratorio de TIA.
He elegido programar un diagnosticador de fallos en la red eléctrica en el encendido de un ordenador personal. Este sistema experto se encarga de diagnosticar el problema en el caso de que un ordenador no sea capaz de arrancar, pudiendo así ser de ayuda en el caso de que un usuario no experto de un ordenador se encuentre en esta tesitura. Cuando el ordenador no arranque, se realizará un análisis del entorno, las posiciones de los interruptores, las conexiones de los cables o el flujo de la electricidad en la habitación, el enchufe, la regleta y finalmente la fuente de alimentación, así como el correcto funcionamiento del botón de encendido.
Para ello, dispondremos de un set de datos observables como la luz indicadora de corriente en la placa base, la luz del botón de la regleta, la luz en la habitación, o el hecho de que el ordenador pueda encenderse. También serán datos observables las conexiones, la del botón de power, la de la regleta al enchufe y la del propio ordenador a la regleta o al enchufe.
Todos estos datos a su vez, forman parte también del set de datos deducibles, por ejemplo si la luz de la regleta está encendida, podremos deducir que hay luz en la habitación y que el enchufe funciona correctamente.
Finalmente, la solución al problema consistirá en el diagnóstico. El problema puede estar desde las conexiones entre componentes hasta en el funcionamiento de hardware del ordenador como la fuente o el botón de encendido como también en objetos externos como la regleta, el enchufe o la propia corriente de la habitación.

Diseño del sistema experto

Para el diseño de este sistema experto he definido una “template” ENTORNO con todos los atributos o slots propios del entorno sobre el que se realizará el diagnóstico. Para el diagnóstico tendremos un set de reglas que permitirán al sistema deducir dónde está el error en el circuito eléctrico.

Algunas de estas reglas son, por ejemplo:

```
(defrule botpow
            (declare (salience 400)(no-loop TRUE))
?entorno<-(ENTORNO (codigo ?cod)(conexion_botpow_placa si)(enciende
no)(luz_corriente_placa si))
        =>  (modify ?entorno (boton_power mal)))
```

Estas reglas se encargan de sacar las conclusiones finales, en este caso el sistema a partir de saber que el boton de power está conectado al ordenador, a este le llega corriente y este no enciende, concluye que el botón de power no está funcionando como toca. Este tipo de reglas estan para obtener la conclusión final del diagnóstico del problema (fuente de alimentación, corriente, regleta, enchufe, etc.).

Otro tipo de reglas son las deductoras, por ejemplo:

```
(defrule deduc_enchuf
               (declare (salience 300))
?entorno<-(ENTORNO (codigo ?cod)(enchufe bien))
        =>  (modify ?entorno (luz_ENTORNO si)))
```

Estas reglas son las encargadas de obtener información a partir de la información existente, sin necesidad de preguntar al usuario. En este caso el sistema deduce que, si el enchufe funciona correctamente quiere decir que tenemos luz en la habitación. Estas reglas tienen menor prioridad que las anteriores ya que si hace matching con una de las anteriores, debe tenerse en cuenta antes que estas ya que el problema está resuelto.
Finalmente tenemos las últimas reglas:

```
(defrule pregunta_luzplaca
        (declare (salience 0))
        (exists (need-ENTORNO (codigo ?cod)(luz_corriente_placa nil)))
?g<- (ENTORNO (codigo ?cod)(luz_corriente_placa nil)(diagnostico nil))
        => (printout t "La luz de la placa que indica corriente esta
encendida? " ?cod "? (responder con si o no)")
           (modify ?g (luz_corriente_placa(read))))
```

Este tipo de reglas tenemos una para todos los atributos observables, en caso de que el sistema no tenga la información o que posteriormente no haya sido capaz de deducirla, siendo estas reglas las de menor prioridad, el sistema recurrirá a preguntar al usuario por el estado de algún atributo. En el ejemplo el sistema pregunta al usuario si la luz de la placa que indica que el ordenador tiene corriente

está encendida, con esta información el sistema será capaz de sacar nuevas conclusiones o de encontrar el problema.

Evaluación con casos de prueba

Para comprobar el correcto funcionamiento del sistema experto he realizado múltiples evaluaciones con diferentes casos de prueba. Aquí analizaré algunos de ellos.

Caso trivial:

```
(deffacts inicio
      (ENTORNO (codigo pc3)(enciende si)))

> Funciona correctamente pc3
```

En este caso el ordenador arranca directamente, por lo tanto no existe ningún tipo de problema en el circuito.

Forward-chaining:

```
(deffacts inicio
      (ENTORNO(codigo pc3)(enciende no)(conexion_botpow_placa
si)(boton_fuente_pos encendido)(luz_corriente_placa no)(conexion
regl)(boton_regleta_pos encendido)(luz_boton_regleta no)(conexion_regl_ench
si)(luz_ENTORNO si)))
```

En este caso, el sistema tiene toda la información necesaria para deducir que el problema de encendido está en el enchufe ya que no está proporcionando corriente a la regleta y por lo tanto no se enciende el botón de encendido.

Deducción mediante inferencia:

```
(deffacts inicio
  (ENTORNO (codigo pc1)(enciende no)(conexion_botpow_placa
si)(luz_corriente_placa si)))
```

En este caso el sistema deduce que al haber corriente en la placa tanto la regleta como el enchufe como la electricidad funcionan correctamente y por lo tanto el fallo existe en el botón de encendido del ordenador.

Backward-chaining:

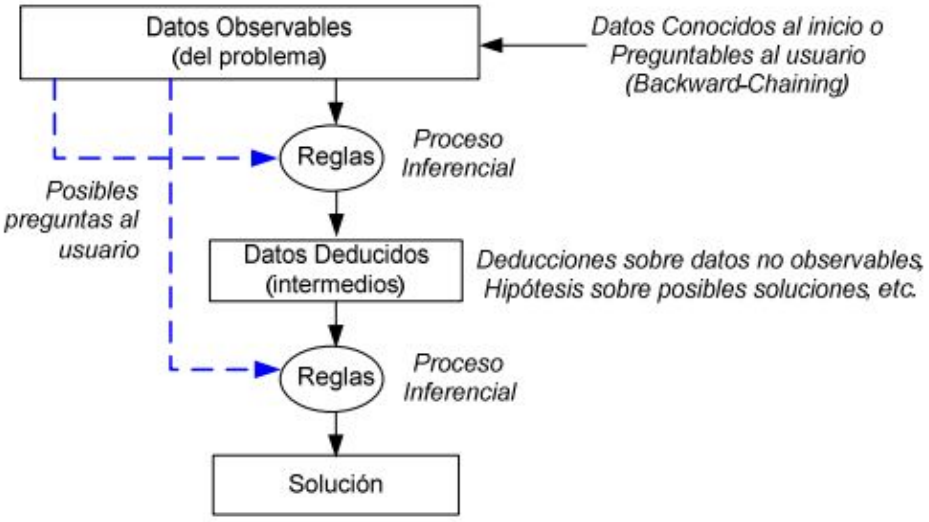
```
(do-backward-chaining ENTORNO)
(deffacts inicio
  (ENTORNO (codigo pc1)(enciende no)(conexion_botpow_placa
si)(boton_fuente_pos encendido)(luz_corriente_placa no)))
```

```
>Donde esta conectado el pc? pc1 (responder con ench(ufe) o regl(eta))regl
>En que posicion esta el boton de la regleta? pc1 (responder con encendido
o apagado) encendido
>Hay luz en la habitacion? pc1 (responder con si o no) no
>El fallo esta en la corriente de la sala pc1
```

Finalmente en este caso observamos como el sistema a falta de información suficiente para deducir el problema lanza una batería de preguntas al usuario mediante las cuales el sistema es capaz de llegar a una conclusión final.

Crítica

Como se ha podido observar en esta última sección, el sistema experto está programado para ser compatible tanto con forward como con backward chaining, tiene su set de datos observables, sus reglas intermedias para deducir datos y las reglas para alcanzar la solución siguiendo el esquema mostrado en el boletín.



De esta manera no es necesario conocer toda la información para alcanzar la solución, mediante deducciones y preguntas podemos agilizar y permitir el proceso que de otra forma sería imposible.