Ingeniería del Conocimiento 2021-2022
Grado en Ingeniería Informática
Universidad de Granada

Diseño de Experto Modular

Autores:
Brian Sena Simons.

Fernando Pastrana Gomez

Grupo:
3ºA Subgrupo A2

1. Idea Conceptual del sistema

La idea por detrás del sistema modular diseñado en esta práctica es el empleo de un sistema de basado en turnos como en una entrevista real. Cada experto tiene un turno para realizar una pregunta y un turno para realizar una inferencia.

Para ello es necesario definir un conocimiento compartido entre "expertos" o modulos en el cual se van guardando las preguntas respondidas y la información obtenida para evitar repetir preguntas. De esta manera todos los modulos pueden mantener su diseño original con apenas cambiando los disparadores de las preguntas para que hagan uso de la información obtenida y un par de funciones de "cambio de estado" para añadir la información obtenida por el otro módulo.

Un ejemplo de reglas necesarias para la ejecución:

```
;; Cambiamos de turno cada dos movimientos
(deffunction change_turn (?x)
    (if
    (eq (fact-slot-value ?x next) true)
    then
        (modify ?x (preguntado false)) (modify ?x (next false)) (focus A)
    else
        (modify ?x (next true))
    )
)
;; Verificar que hemos terminado para ir al modulo imprimir
(defrule TERMINADOS
(declare (salience 9999))
    ?e <- (estados
        (estadoA TERMINADO)
        (estadoB TERMINADO)
```

```
      )
    =>
    (focus IMPRIMIR)
)
;; Solo en caso de que no se ejecute por alguna raz n el cambio
(defrule not_changed
    (declare (salience -9999))
    ?t <- (next-turn (next ?x))
    (not (estados
                (estadoA TERMINADO)
                (estadoB TERMINADO) )
    )
    =>
    (change_turn ?t)
)
```

En esas funciones se pueden observar el comportamiento de cambio de turnos utilizado por nuestro método. A continuación se observa un ejemplo de disparador utilizando el conocimiento compartido:

```
(defrule preguntar_software
    (declare (salience 5))
    ;; Conocimiento compartido
    ?r <- (Respuestas (software UNKNOWN))
    ?t <- (next-turn (preguntado false))
    ;; Conocimiento del k - experto
    ?f <- (software UNKNOWN)
    (not (finished) )
    =>
    [......]
)
```

2. Ejemplos de Ejecución

Podemos ver a continuación un ejemplo de ejecución de nuestro sistema modular capaz de realizar correctamente el cambio de turno en la figura 2.1 y en la figura 2.2 un ejemplo de salida del prototipo 2B.

```
5 ramas en diferentes categorias, empezemos!
4 Si tuvieras que elegir, ¿dirías que prefieres
3 Software a Hardware? (SI NO NOLOSE): SI
2 Sabia que eras uno de los mios! ;)
1 ¿Como se te dan las matematicas? (bien,mal,regular,no_se)
```

Figura 2.1: Primera pregunta del Experto 1(4) y segunda del Experto 2(1)

```
0 ################## Experto 1 ##################
9 El experto 1 dice que: Ingenieria de Software,
8 El experto 1 justifica que: eres más de ponerle a prueba lo que te enseñan, te
7 gustan las matematicas, no tienes la nota tan alta
6 pero te esfuerzas bastante
5 ##############################################
4 ################## Experto 2 ##################
3 El experto 2 dice que: Ingenieria de Computadores
2 El experto 2 justifica que: te encanta el hardware
1 ##############################################
```

Figura 2.2: Resultado de una cierta combinación de respuestas al sistema

El diseño es escalable hasta k-expertos iempre y cuando se modifiquen correctamente el conocimiento compartido por todos, sus disparadores y sobre todo la regla de intercambio entre módulos y una función de terminación que espere a todos. La clave está en apenas llamar a la función cambiar turno cuando se haya inferido a traves de una pregunta.

Referencias