Grado en Ingeniería Informática, Universidad de Córdoba

Sistemas Inteligentes

CLIPS

Tema 4: Reglas I

Aurora Esteban Toscano
aestebant@uco.es

21 de marzo de 2022

Objetivos

- Introducción a las reglas en CLIPS.
- Conocer el ciclo básico de ejecución de reglas.
- Profundizar en la definición y las propiedades de las reglas.

Introducción a las reglas en CLIPS

Si el **antecedente** es cierto según los hechos almacenados en la **base de afirmaciones**, entonces **pueden** realizarse las acciones especificadas en el **consecuente**.

Estructura de las reglas

Antecedente $\Longrightarrow$ **Consecuente**

- Antecedente: cero o más *cláusulas* que deben cumplirse para que la regla pueda ejecutarse (dispararse).
- Consecuente: cero o más *acciones* que se llevarán a cabo si la regla se dispara.
  - Entre esas acciones puede estar crear (afirmar) más hechos, eliminar hechos obsoletos, llegar a conclusiones finales...

Por ejemplo

`SI` voy a clase `Y` atiendo `Y` hago los ejercicios $\Longrightarrow$ apruebo la asignatura

Gestión de reglas

Elementos de los Sistemas Expertos relacionados con la gestión de reglas:

- Base de conocimiento: conjunto de reglas que describen el problema a resolver.
- Motor de inferencia: comprueba el antecedente de las reglas y la base de afirmaciones, decide la próxima regla a ejecutar y aplica su consecuente.
- Activación o disparo de reglas: las reglas se disparan una única vez para un mismo conjunto de antecedentes $\Rightarrow$ evitar entrar en bucles.
  - Entidad patrón: elementos en los que se basa la activación de una regla. Pueden ser hechos ordenados, plantillas e instancias de clases.
  - Elemento condicional (EC): son cada una de las condiciones que pueden encadenarse para componer el antecedente de la regla.

Definición de las reglas I

Constructor

```
(defrule <nombre> [comentario] [propiedades]
     <elemento-condicional>*
     =>
     <acción>*
)
```

- Las reglas deben tener nombre único: si se define una regla con el mismo nombre que otra ya existente, la sobrescribirá.
- No hay límite en el número de EC ni de acciones que una regla tenga.
  - Si no hay ningún EC en el antecedente ⇒ la regla se ejecuta automáticamente.
  - Si no hay ninguna acción en el consecuente ⇒ ejecución sin consecuencias.

Definición de las reglas II

- Las reglas se **activan** cuando se satisface su antecedente.
  - Las reglas pueden activarse para distintos conjuntos de hechos → cada activación es una *instancia de regla*.
- Las instancias de reglas se almacenan en la **agenda**: muestra las reglas disponibles para ejecutar.
  - Para cada instancia de regla muestra: la prioridad de la regla, el nombre y los índices de los hechos que la satisfacen.
- Las reglas se **disparan** cuando el motor de inferencia decide ejecutar las acciones de sus consecuentes.
- Se puede fijar la **prioridad** de una regla entre [-10000, 10000]. Prioridad por defecto: 0.
- La **estrategia de resolución de conflictos** decide qué regla disparar la siguiente si hay varias con la misma prioridad.

Ejemplo

```
CLIPS> (clear)
CLIPS> (defrule hola1 => (printout t "Hola Mundo" crlf))
CLIPS> (defrule hola2 (initial-fact) => (printout t "Hola Mundo iniciado" crlf)))
CLIPS> (rules)
hola1
hola2
CLIPS> (agenda)
0      hola1: *
CLIPS> (assert (initial-fact))
CLIPS> (agenda)
0      hola2: f-1
0      hola1: *
CLIPS> (run)
Hola Mundo iniciado
Hola Mundo
CLIPS> (facts)
f-1     (initial-fact)
CLIPS> (agenda)
CLIPS>
```

Ciclo básico de ejecución de reglas

1. Las reglas activadas se disparan con el comando (`run [pasos]`)
2. Si se ha alcanzado el máximo de pasos $\Rightarrow$ fin de la ejecución.
3. Se actualiza la agenda según la lista de hechos.
4. Se selecciona la próxima instancia de regla a ejecutar según prioridades y estrategia de resolución de conflictos.
5. Se dispara la instancia seleccionada, se incrementa el número de pasos y se elimina de la agenda.
6. Se vuelve al paso 2.

Ejemplo: animales.clp

El ejercicio muestra una serie de reglas que, en base a un hecho, indican qué sonido hace un animal.

1. Descarga de Moodle el fichero y observa la estructura.
2. Carga el fichero en CLIPS:
   1. Establece el directorio de trabajo a la ruta donde esté el fichero guardado. `Environment > Set Directory...`
   2. Carga el fichero ejecutando (`load animales.clp`)
   3. Inicializa la base de afirmaciones.
3. ¿Cómo se ampliaría la base de conocimiento para contemplar otros animales como ovejas, vacas o patos?
4. ¿Cómo se adaptarían las reglas para trabajar con hechos no ordenados?

Ejemplo: coche.clp

El ejercicio muestra cómo se puede actualizar la información de la base de hechos por medio de reglas, tanto sobre hechos ordenados como no ordenados.

1. Descarga de Moodle el fichero y observa la base de conocimiento (dos reglas) y la base de afirmaciones (dos hechos).
2. ¿Cuál será el flujo de inferencia?
3. Carga el fichero en CLIPS e inicializa la base de afirmaciones.
4. Ejecutar el motor de inferencia en pasos de uno en uno y ver el resultado.

Propiedades de las reglas

Definición

```
(defrule <nombre> [comentario]
    (declare <propiedad>)
    <antecedente> => <consecuente>
)
```

- Las propiedades permiten alterar el comportamiento en la activación de las reglas.
- La declaración de propiedades se incluye en la definición de la regla, después del comentario y antes del antecedente mediante la sentencia (`declare` …).
- Una regla puede tener una sola sentencia (`declare`).

Propiedades de las reglas: prioridad

```
(declare (salience <prioridad>))
```

- La prioridad determina en qué orden se disparan varias reglas activadas.
- Puede variar en el rango [-10000, 10000] $\rightarrow$ a mayor número, mayor prioridad y antes se dispara la regla.
- Valor de prioridad por defecto: 0.
- Evaluación de la prioridad
  - Prioridad estática: en la definición de la regla. Comportamiento por defecto.
  - Prioridad dinámica: (1) cuando se activa la regla o (2) en cada ciclo de ejecución.

Ejemplo

```
CLIPS> (clear)
CLIPS> (defrule regla1 (declare (salience 10))
   (initial-fact) => (printout t "Me ejecuto la primera" crlf)
)
CLIPS> (defrule regla2 (declare (salience -10))
   (initial-fact) => (printout t "Me ejecuto la segunda" crlf)
)
CLIPS> (assert (initial-fact))
CLIPS> (agenda)
10     regla1: f-1
-10    regla2: f-1
CLIPS> (run)
Me ejecuto la primera
Me ejecuto la segunda
```

Ejercicio pila.clp

1. Descarga de Moodle el fichero `pila.clp`, que implementa una estructura de datos tipo *pila*.
2. Observa la estructura de las operaciones *push* y *pop* y su comportamiento una vez se ejecuta el programa.
3. ¿Por qué una regla tiene más prioridad que otra?
4. ¿Cómo se implementaría una estructura de datos tipo cola?
5. Si en la misma base de conocimiento coexisten las dos estructuras, ¿cómo indicamos al SE cuál de ellas queremos utilizar?

Otras gestiones sobre reglas

- (list-defrules): listar las reglas definidas.
- (ppdefrule <nombre>): muestra la definición de una regla dada.
- (undefrule <nombre>): suprime la regla dada.

Además, de recordatorio...

- (load <fichero.clp>): carga un script en CLIPS.
- (facts): muestra la base de hechos o afirmaciones.
- (rules): muestra la base de conocimiento.
- (agenda): muestra la agenda.
- (reset): borra la base de hechos, reinicia los hechos iniciales y reinicia la agenda.
- (clear): borra toda la información del sistema.
- (exit): salir de CLIPS.

Grado en Ingeniería Informática, Universidad de Córdoba

Sistemas Inteligentes

CLIPS

Tema 4: Reglas I

Aurora Esteban Toscano
aestebant@uco.es

21 de marzo de 2022