Grado en Ingeniería Informática, Universidad de Córdoba

Sistemas Inteligentes

CLIPS

Tema 6: Módulos

Aurora Esteban Toscano
aestebant@uco.es

28 de marzo de 2022

Objetivos

- Conocer la estructuración en *módulos* de CLIPS.
- Extender el conocimiento del ciclo de ejecución a la presencia de varios módulos.
- Dirigir el ciclo de ejecución entre módulos.

Introducción a los módulos en CLIPS

Los módulos en CLIPS permiten facilitar el control del razonamiento de forma modular agrupando constructores (hechos, reglas, etc.)

Definición

```
(defmodule <nombre> [comentario]
     <(export <port-item)>*
     <(import <modulo> <port-item>)>*
)
<port-item>:: ?ALL | ?NONE | <constructor> ?ALL | <constructor> ?NONE |
         <constructor> <nombre>
<constructor>:: deftemplate | defclass | defglobal | deffunction | deffgeneric
```

Propiedades de los módulos

- Siempre existe un módulo por defecto `MAIN`.
- El *foco* determina cuál es el *módulo activo*, es decir, el modo en ejecución.
- Los módulos sólo pueden borrarse con la orden `(clear)`
- Para especificar a qué módulo pertenece un elemento de CLIPS, dos aproximaciones:
  - Referencia explícita: `<módulo>::<elemento>`.
  - Referencia implícita: establecer el *foco* en el módulo previamente, con lo que todos los elementos para los que no se especifique lo contrario se asumirán que están en el *módulo activo*.
- `(get-current-module)`: muestra el módulo actual.
- `(set-current-module <módulo>)`: cambia al módulo especificado.

Ejemplo

```
CLIPS> (clear)
CLIPS> (defmodule A)
CLIPS> (assert (dato 55))
<Fact-1>
CLIPS> (defmodule B)
CLIPS> (assert (dato 33))
<Fact-2>
CLIPS> (defrule busca55
(dato 55) => )
CLIPS> (facts)
f-2     (dato 33)
For a total of 1 fact.
```

```
CLIPS> (agenda)
CLIPS> (list-defmodules)
MAIN
A
B
For a total of 3 defmodules.
CLIPS> (set-current-module A)
B
CLIPS> (facts)
f-1     (dato 55)
For a total of 1 fact.
CLIPS> (rules)
CLIPS>
```

Importación y exportación de elementos

Los elementos exportables entre módulos se tienen que definir:

1. Como exportables en el módulo de origen.
2. Como importables en el módulo de destino.

Posibles ejemplos:

- `(defmodule A (export ?ALL)) / (defmodule B (export ?NONE) (import A deftemplate x))`
- `(defmodule A (export deftemplate ?ALL)) / (defmodule B (import A deftemplate ?ALL))`
- `(defmodule A (export ?ALL)) / (defmodule B (export ?ALL)) / (defmodule C (import A deftemplate datoA) (import B deftemplate datoB))`

OjO

Los hechos de un módulo pueden exportarse a otro mediante su `deftemplate`. Las operaciones que se hagan sobre un hecho tienen efecto en todos los módulos en los que esté.

Ciclo de ejecución con módulos I

- Cada módulo tiene su propia **base de conocimientos** y su propia **agenda** no compartible con otros.
- `(reset)`: restablece las bases de hechos de todos los módulos y devuelve el foco al módulo por defecto `MAIN`.
- `(clear)`: restablece completamente el sistema y devuelve el foco al módulo por defecto `MAIN`.
- `(run)` ejecuta la agenda del módulo activo $\rightarrow$ las instancias de sus propias reglas que se hayan activado para su propia base de hechos.
- Los módulos se organizan en una pila: conforme un módulo acaba pasa al siguiente de la pila $\rightarrow$ el programa termina cuando no quedan módulos activos.
  - `(get-focus-stack)`: muestra el estado actual de la pila de módulos activos.

Ciclo de ejecución con módulos II

Formas de alterar el ciclo de ejecución normal:

- Orden `focus` en el consecuente de una regla: incluye un nuevo módulo en la pila de módulos activos y pasa el control a la agenda de ese módulo.
- Orden `return` en consecuente de una regla: termina antes de tiempo con la agenda del módulo actual para pasar el control a la del siguiente módulo activo en la pila.

Ejemplo

- Descarga de Moodle el fichero de fichero `modulos1.clp`. Observa la estructura y cárgalo en CLIPS. Ve ejecutando la agenda de uno en uno, atendiendo a cómo cambian las bases de afirmaciones y el foco del programa.
- Descarga de Moodle el fichero de fichero `modulos2.clp`. Observa la estructura y cárgalo en CLIPS. Ve ejecutando la agenda de uno en uno, atendiendo a cómo cambian las bases de afirmaciones y el foco del programa.

Otras gestiones sobre módulos

- (list-defmodules): muestra todos los módulos definidos en el sistema.
- (ppdefmodule <módulo>): muestra la definición de un módulo dado.

Grado en Ingeniería Informática, Universidad de Córdoba

Sistemas Inteligentes

CLIPS

Tema 6: Módulos

Aurora Esteban Toscano
aestebant@uco.es

28 de marzo de 2022