Введение в CLIPS

Оболочка для разработки экспертных систем – C Language Integrated Production System

Язык программирования CLIPS

- CLIPS – среда (система) разработки ЭС.
- Определяет язык программирования, описанный в Clips Basic Programming Guide.
- Поддерживает программирование на правилах (с использованием встроенного алгоритма вывода – RETE), процедурное, ООП.
- Преемник OPS, начало разработки – 1984.
- Актуальная версия CLIPS 6.3, выпущена 17 марта 2015.
- Не поддерживает обратный вывод (в оригинале).

Основные элементы CLIPS

- База знаний – представлена в виде **фактов**, имеет атрибуты слотового представления.
- Начальные знания представляются в виде некоторого набора исходных фактов.
- Факты образуют список, по умолчанию присутствует нулевой.
- Список правил, шаблоны правил.
- Машина логического вывода, осуществляющая прямой вывод (forward chaining) алгоритомом RETE.

Язык CLIPS

- Имеет общее название COOL.
- Тесная интеграция с C, возможность компиляции и использования функций, написанных на C.
- Синтаксис похож на LISP, заимствования из Ada, Pascal и др.
- Разрабатывался в Космическом центре NASA/Джонсон, там же большая часть упоминаемых систем на его основе была реализована, пока в 1995 не свернули финансирование.
- До пятой версии ориентирован на правила, дальше добавили поддержку процедурного программирования, ООП и проч.
- Имеет простейшую IDE для работы, а также интерпретаторы для некоторых ОС.

Основы

- Case-sensitive, нет поддержки кириллицы.
- Восемь типов полей – вещественное, целое, символ, строка, внешний адрес, адрес факта, имя экземпляра, адрес экземпляра.
- Символы – некоторые последовательности, из печатных символов (0 или более), Lazy-students_are-!?#$^B012389, символы ? и $ в начало символа не ставить.
- Команды – в круглых скобках, иначе это интерпретируется как символ.
- Строки – в двойных кавычках.

Базовые команды

`(load <file-name>)` – загрузка из файла. Функция, по умолчанию отображает процесс, TRUE/FALSE.

`(save <file-name>)` – сохранение в файл текущих конструкций (environment).

`bsave/bload` – аналогично, двоичный файл.

`(clear)` – очистка окружения. Некоторые элементы не удаляются/не изменяются.

`(exit [<integer-expression>])` – выход.

`(reset)` – Очищает список фактов, целей, объекты (instances), реинициализация, смена модуля на MAIN.

`(agenda)` – текущие правила, готовые к активации

Определение фактов

```
(deftemplate car_problem
   (slot name)
   (slot status)
)
(deffacts trouble_shooting
   (car_problem (name ignition_key) (status on))
   (car_problem (name engine) (status wont_start))
   (car_problem (name headlights) (status work))
)
(defrule rule1
   (car_problem (name ignition_key) (status on)) (car_problem
   (name engine) (status wont_start))
=>
   (assert (
        car_problem (name starter) (status faulty)))
)
```

Определение фактов

Определяем шаблон факта:

```
(deftemplate <relation-name> [<optional-comment>]
   <slot-definition>*)
<slot-definition> = (slot <slot-name>) |
(multislot <slot-name>)
```

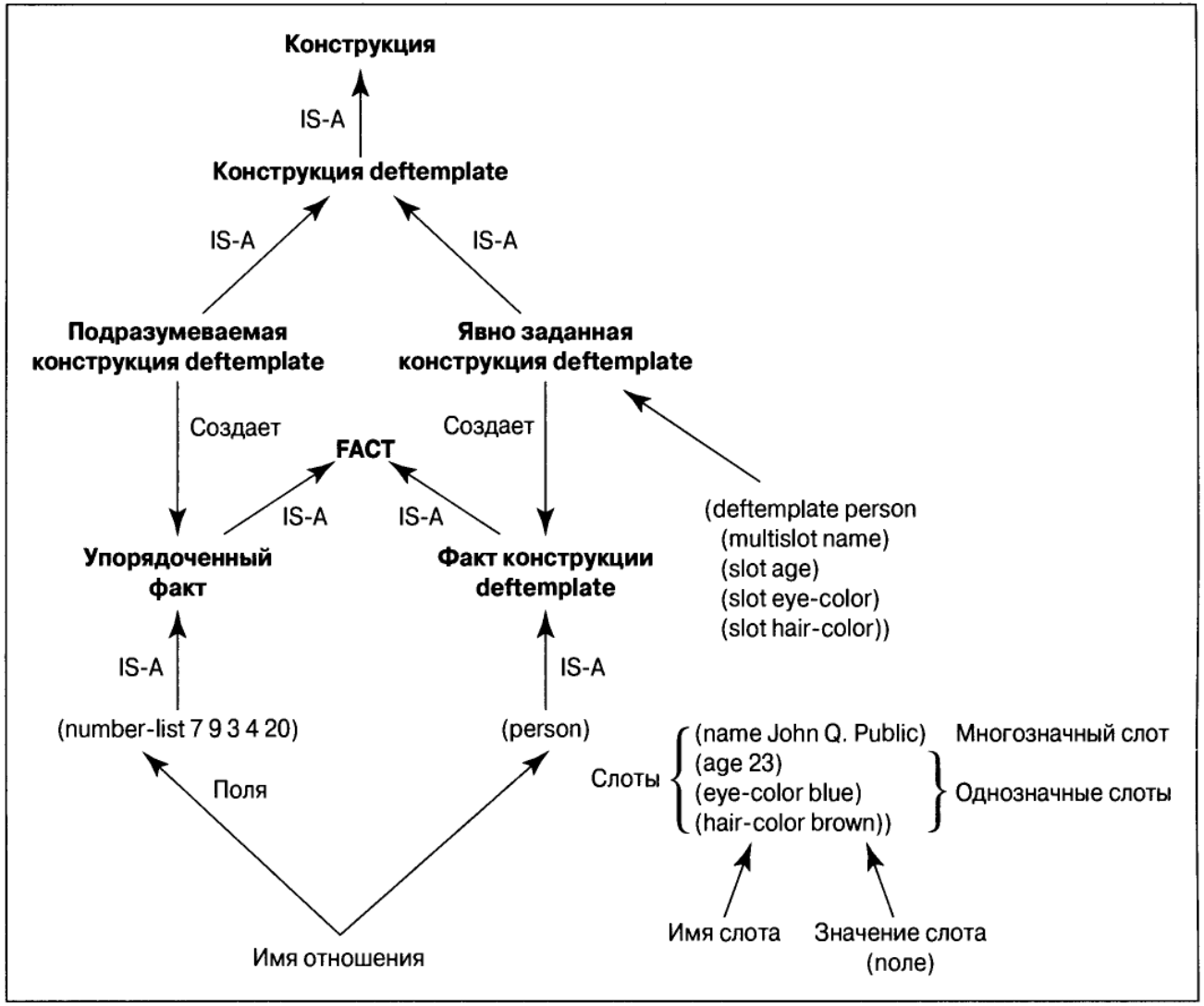
Факты делятся на факты с конструкцией deftemplate (non-ordered facts) и упорядоченные факты (ordered facts). Порядок слотов не важен.

Многозначные слоты могут хранить несколько (0 и более) значений.

(Person (name Albus Dumbledore) (age 112)) – в каком случае такое описание факта допустимо?

Просмотреть текущую коллекцию фактов можно командой (facts), сбросить – (reset)

Виды фактов


Конструкция
IS-A
Конструкция deftemplate
IS-A
IS-A
Подразумеваемая конструкция deftemplate
Явно заданная конструкция deftemplate
Создает
Создает
FACT
IS-A
IS-A
Упорядоченный факт
Факт конструкции deftemplate
(deftemplate person
(multislot name)
(slot age)
(slot eye-color)
(slot hair-color))
IS-A
IS-A
(number-list 7 9 3 4 20)
(person)
Поля
Слоты
(name John Q. Public)
(age 23)
(eye-color blue)
(hair-color brown))
Многозначный слот
Однозначные слоты
Имя слота
Значение слота (поле)
Имя отношения

Список фактов

Добавление фактов происходит командой assert:

`(`**`assert`** `<fact>+)`

Просмотреть текущую коллекцию фактов можно командой (facts), сбросить – `(reset)`

Каждому факту система присваивает индекс, не обязательно последовательный.

По умолчанию ввод дублирующих фактов не производится – выполнение такой команды результата не имеет.

Извлечение фактов – команда `(`**`retract`** `<fact-index>+)`**`.`**

Правила

(**defrule** <rule-name> [<comment>]
[<declaration>] ; Rule Properties
<conditional-element>* ; Left-Hand Side (LHS)
=>
<action>*) ; Right-Hand Side (RHS)

Right-hand side и **left-had side** – блоки. По умолчанию **and**.

Salience – приоритет правил.

Могут изменять текущую повестку, перемещать фокус по модулям.

Порядок работы

Agenda – список правил, для которых были выполнены все условия (готовых к активации), и которые ещё не были выполнены. У каждого модуля своя, аналог стека.

1. Если предел активаций не достигнут, то выбирается первое правило из Agenda. Если пуст – переход по фокусу.
2. Для выполненного правила активируется RHS, изменяющая как повестку, так и фокус.
3. Agenda перестраивается с учётом приоритета (salience) и стратегии разрешения конфликтов.
4. По завершению – выход.

Литература

1. Рассел С., Норвиг П. Искусственный интеллект. Современный подход, 2-е изд., – М.: «Вильямс», 2006.
2. Люгер Дж. Ф. Искусственный интеллект. Стратегии и методы решений сложных проблем, 4-е изд., – М.: «Вильямс», 2003.