Среда CLIPS

Мадорский Константин

ВМК МГУ
2013

Среда Clips

- CLIPS — интегрированная с языком Си среда построения продукционных систем (C Language Integrated Production System).
- CLIPS является одной из наиболее широко используемых инструментальных сред для разработки экспертных систем благодаря своей скорости, эффективности и бесплатности.
- Управляющие стратегии — встроены.
- Достаточно задать правила и факты.
- CLIPS включает полноценный объектно-ориентированный язык *COOL* для написания экспертных систем.

Типы данных

- **float** ; 237e3 15.09 +12.0 -32.3e-7
- **integer** ; 237 15 +12 -32
- **symbol** ; Hello B76-HI bad_value 127A @+=-%
- **string** ; "a and b" "1 number“ "a\”quote\\b"
- **external-address** ; адрес структуры данных,
  ; возвращенной внешней функцией
- **fact-address** ; индекс факта
- **instance-name** ; имя объекта, т.е. экземпляра
  ; класса, определенного пользователем
  ; [obj_name] [+++] [123-890]
- **instance-address** ; ссылка на объект

Типы функции

- Определенные пользователем внешние
- Системные (внутренние)
- Определенные с помощью конструктора **deffunction**
- родовые функции

Конструкторы

- deffacts
- deftemplate
- deffunction
- defrule
- defglobal
- defclass
- definstances
- defmessage-handler
- defgeneric
- defmethod
- defmodule

Факты

- Могут быть неупорядоченными и упорядоченными.
- Конструкторы: **deftemplate** и **deffacts**
- Функции: **assert**, **retrect**, **modify**, **duplicate**, **assert-string**, **fact-existp**
- Функции только для неупорядоченных фактов: **fact-relation**, **fact-slot-names** и **fact-slot-value**

Правила

- Конструктор создания правил **defrule**
- Свойства правила **salience** и **auto-focus** задаются с помощью функции **declare**
- Стратегии: глубины (depth), ширины (breadth), упрощения (simplicity), усложнения (complexity), LEX, MEA, случайная (random)

Пример программы на CLIPS

Задача состоит в том, чтобы создать 5 стрел.

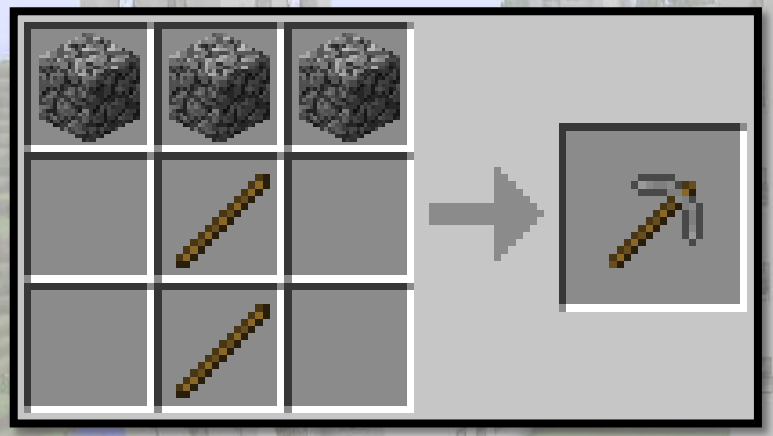
- **Создание одной стрелы (arrow):**

flint + stick + feather -> arrow

- **Палки (stick) и перья (feather) мы можем получать просто так.**

Пример программы на CLIPS

- Кремни (flint) необходимо добыть с использованием киркомотыги (pickaxe), которую в свою очередь необходимо создать:

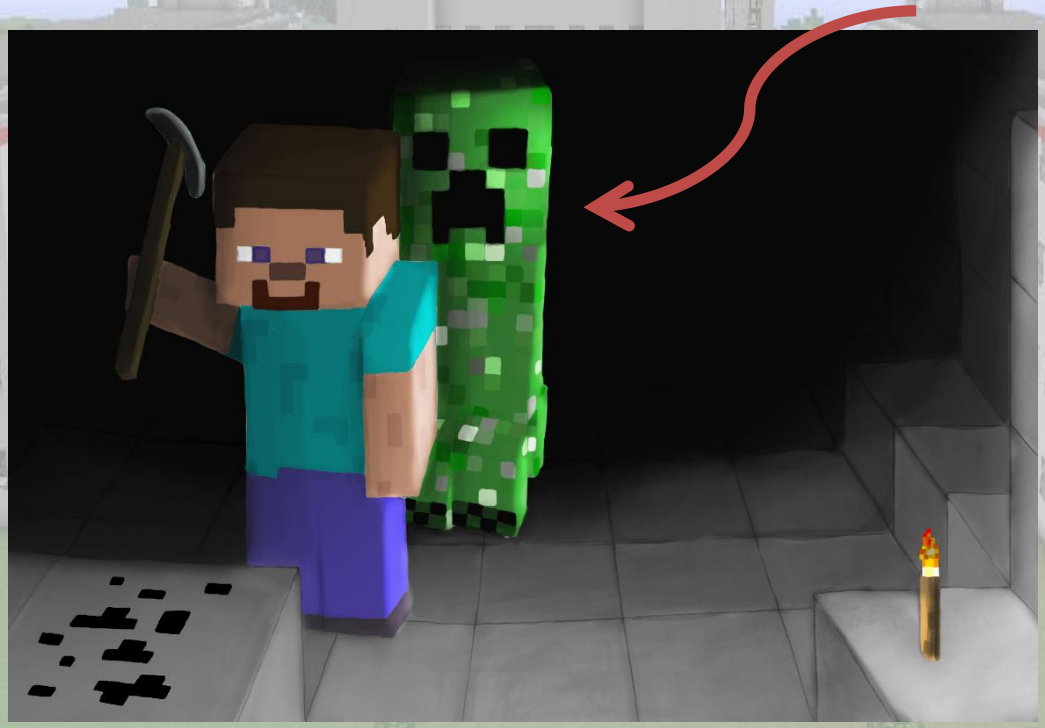


3*cobblestone +
2*stick -> pickaxe

- Булыжники (cobblestone) так же можно получать просто так.
- Киркомотыга ломается после 10 ударов!

Пример программы на CLIPS

- Добыча кремня с большой вероятностью приведет к получению булыжника, с меньшей вероятностью – к получению кремня и с совсем малой вероятностью – к смерти от взрыва крипера.



Пример программы на CLIPS

- Можно использовать «чит-коды»:
  (modify ?F (arrow 999999)),
  где вместо ?F нужно ввести номер факта рюкзака.

Спасибо за внимание!

И спользуемая литература:

«Разработка экспертных систем. Среда CLIPS.»
А.П. Частиков, Т.А. Гаврилова, Д Л. Белов

; для запуска программы (to start the program):

;(set-strategy breadth)

; Click "Window" -> "1 Facts" to watch how facts change

;(clear)

;(load arrow.clp)

;(reset)

; "Ctrl + R" - to start program until (halt)

; try to press it several times

; "Ctrl + T" - to make one step

```
; шаблон киркомотыги
(deftemplate pickaxe "this comment will stay"
        (slot quantity (type INTEGER) ) ;this one won't
        (slot hits_left (type INTEGER) )
)

; шаблон рюкзака
; в рюкзаке лежат стрелы, булыжники, кремни, палки, перья
(deftemplate backpack
        (slot arrow (type INTEGER) )
        (slot cobblestone (type INTEGER) )
        (slot flint (type INTEGER) )
        (slot stick (type INTEGER) )
        (slot feather (type INTEGER) )
)

(deffacts initial-facts-set   ;initializes after each (reset)
        (pickaxe (quantity 0) (hits_left 0))
        (backpack (arrow 0)
                (cobblestone 5)
                (flint 0)
                (stick 4)
                (feather 10) )
)
```

```
; функция для получения рандома при добыче кремня
(deffunction get_rand ()
		(bind ?x (mod (+ (time) (random)) 100))
		(if (< ?x 30) then
				(return 0)  ; +1 flint
		else (if (< ?x 98) then
					(return 1)  ; +1 cobblestone
				else (return 2)  ; death caused creeper explosion
				)
		)
)

; правило создания стрелы
(defrule craft_arrow
		(declare (salience 10))
		?f1 <- (backpack (arrow ?A) (flint ?FL) (stick ?S) (feather ?F))
		(test (and (>= ?FL 5) (>= ?S 5) (>= ?F 15) ) )
=>
		(modify ?f1 (arrow (+ ?A 5)) (flint (- ?FL 5)) (stick (- ?S 5))  (feather (- ?F 15)) )
		(printout t crlf "Finally you've created five arrows!" crlf crlf)
		(halt)
)
```

```
; правило создания киркомотыги
(defrule craft_pickaxe
        (declare (salience -5))
        ?f1 <- (pickaxe (quantity ?Q))
        ?f2 <- (backpack (stick ?S) (cobblestone ?C) (flint ?FL))
        (test (< ?FL 5)) ; мотыга не нужна, если уже добыто 5 кремней
        (test (< ?Q 1)) ; нам хватит и одной киркомотыги
        (test (>= ?S 2)) ; материл необходимый для создания
        (test (>= ?C 3)) ; материл необходимый для создания
=>
        (modify ?f1 (quantity (+ ?Q 1)))
        (modify ?f2 (stick (- ?S 2)) (cobblestone (- ?C 3)))
)
```

```
; правило добычи кремня
(defrule get_flint
        ?f1 <- (pickaxe (quantity ?Q) (hits_left ?HL))
        ?f2 <- (backpack (flint ?FL) (cobblestone ?C))
        (test (< ?FL 5)) ; нет нужды в более чем 5 кремнях
        (test (or (> ?Q 0) (> ?HL 0)))
=>

        ; ВАЖНО не использовать в правиле более одного modify
        ; для одного факта, иначе необходимо отследить какой
        ; номер получил факт после применения первого modify!

        (bind ?r (get_rand))
        (if (= ?r 2) then (modify ?f2 (arrow 0)
                                      (cobblestone 0)
                                      (flint 0)
                                      (stick 0)
                                      (feather 0) )
                          (modify ?f1 (quantity 0) (hits_left 0))
                          (printout t crlf "BOOM!!! You've been killed by creeper explosion..." crlf crlf)

        else
                        (if (> ?HL 0) then ?f1 <- (modify ?f1 (hits_left (- ?HL 1)))
                        else (if (> ?Q 0) then ?f1 <- (modify ?f1 (quantity (- ?Q 1))
                                                                  (hits_left 9))
                              )
                        )
                        (if (= ?r 0) then (modify ?f2 (flint (+ ?FL 1)))
                        else (modify ?f2 (cobblestone (+ ?C 1)))  )
        )
)
```

```
; правило получания палки
(defrule get_stick
		(declare (salience -10))
		?f <- (backpack (stick ?X))
		(test (< ?X 5)) ;нет нужды в более чем 5 палках
=>
		(modify ?f (stick (+ ?X 1)))
)

; правило получания трех пера
(defrule get_feather
		(declare (salience -15))
		?f <- (backpack (feather ?X))
		(test (< ?X 15)) ;нет нужды в более чем 15 перьях
=>
		(modify ?f (feather (+ ?X 3)))
)

; правило получения булыжника
(defrule get_cobblestone
		(declare (salience -10))
		?f <- (backpack (cobblestone  ?X))
		(test (< ?X 3)) ; трех булыжников хватит
=>
		(modify ?f (cobblestone (+ ?X 1)))
)
```