Общая характеристика среды CLIPS

CLIPS (C Language Integrated Production System) поддерживает **три** основных способа представления знаний:
- продукционные правила (для эвристических знаний),
- функции для представления процедурных знаний,
- объектно-ориентированное программирование,

и **шесть** основных черт ООП: классы, обработчики сообщений, абстракции, инкапсуляция, наследование и полиморфизм.

Правила могут сопоставляться с **объектами** и **фактами**. Приложения могут разрабатываться с использованием только правил, только объектов или их комбинации.

Объекты могут использоваться без правил путем посылки сообщений и в этом случае отпадает необходимость в машине вывода, если используются только объекты.

Базовые типы данных и представление фактов

Поддерживаются **восемь** базовых типов данных: 1) *целые* (integer) и 2) *вещественные* (float) числа, 3) *символьные* (symbol) и 4) *строковые* (string) данные, 5) *внешний адрес* (external-address), 6) *адрес факта* (fact-address), 7) *имя экземпляра* (instance-name) и 8) *адрес экземпляра* (instance-address).

1)27; +125; -38

2)12.0; -1.59; 237e3; -32.3e-7

3)bad_value; 456-93-039; @+=%

ограничители–неотобр.символ ‘ ( ) & | < ~ **;** не может начинаться с ? и $?

4)"abc"; "a & b"; "a\"quote"; "fgs\\85"

5)адрес внешней структуры данных: <Pointer-XXXXXX>

6)для ссылки на факты: <Fact-XXX>

7)для ссылки на экземпляры классов: [pump-1]; [foo]; [123-890]

8)<Instance-XXX>, где XXX – имя экземпляра

Базовые типы данных, представление фактов (2)

Место, занимаемое одним значением базового типа данных, называется *полем* (field). Значения базовых типов являются одноместными.

Многоместные значения: (a 123); (); (x 3.0 "red" 567)

Упорядоченные факты: (высота 100); (студент Сидоров); (отец Иван Петр); (однокурсники Иванов Петров Сидоров)

Команды: assert – добавляет факт в факт-список; retract – удаляет факт из списка; modify – модифицирует список; duplicate – дублирует факт:

(assert (length 150) (width 15) (weight “big”))

Идентификатор факта (fact-identifier): f-10 ссылается на факт с индексом 10.

Базовые типы данных, представление фактов (3)

Задание исходного множества упорядоченных фактов:
(deffacts <*имя_группы_фактов*> ["<*комментарий*>"] <*факт*>*)

Пример: (deffacts stud "Студент"
(student name John)
(student spec “COMPUTER”))

<u>Неупорядоченные факты</u> - список взаимосвязанных именованных полей – *слотов*. Возможен доступ к полям по именам.
Одиночные (одно поле) и мультислоты (любое число полей).

Базовые типы данных, представление фактов (4)

Шаблоны – для спецификации состава неупорядоченных фактов.

<u>Синтаксис конструкции deftemplate:</u>

(deftemplate <имя шаблона> [“<комментарий>”]
<определение слота-1>
. . .
<определение слота-N>)

<u>Пример шаблона:</u>

(deftemplate object “Шаблон объекта”
(slot name)
(slot location)
(slot weight))

<u>Пример неупорядоченного факта на основе этого шаблона:</u>

(object (name car) (location 100) (weight 600))

Представление правил в базе знаний. Типы условных элементов.

(**defrule** <имя_правила> ["<комментарий>"][<объявление>]
<условный элемент>*; Левая часть правила (антецедент)
=>
<действие>*) ; Правая часть правила (консеквент)

<u>Пример:</u>

```
(defrule R1
(days 2)
(works 100)
=>
(printout t crlf "Свободного времени нет" crlf)
(assert (freetime "no")))
```

Представление правил в базе знаний. Типы условных элементов

Антецедент правила состоит из последовательности условных элементов (УЭ). Если все УЭ правила удовлетворяются при текущем состоянии базы данных, то правило помещается в список готовых к выполнению правил – **агенду**.

Шесть типов условных элементов:

1) УЭ-образцы (Pattern Conditional Elements);
2) УЭ-проверки (Test Conditional Elements);
3) УЭ “ИЛИ” (Or Conditional Elements);
   УЭ “И” (And Conditional Elements);
   УЭ “НЕ” (Not Conditional Elements);
4) УЭ “Существует” (Exists Conditional Elements);
5) УЭ “Для всех” (Forall Conditional Elements);
6) Логические УЭ (Logical Conditional Elements).

Представление правил в базе знаний. Типы условных элементов (1)

УЭ-образец состоит из совокупности ограничений на поля, масок (wildcards) полей и переменных, используемых при сопоставлении УЭ с образцом – фактом или экземпляром объекта.

В УЭ-образцах используются следующие конструкции:

- литеральные ограничения (Literal Constraints);
- одно и многоместные маски (Single- and Multifield Wildcards);
- одно и многоместные переменные (Single- and Multifield Variables);
- ограничения со связками (Connective Constraints);
- предикатные ограничения (Predicate Constraints);
- ограничения возвращаемым значением (Return Value Constraints).

Представление правил в базе знаний. Типы условных элементов (1)

1) Литеральное ограничение

Упорядоченный: (data 1 one “two”)

Неупорядоченный: (person (name Bob) (age 20))

2) Одно- и многоместные маски:

(data ? blue red $?) будет сопоставляться со следующими упорядоченными фактами:

(data 1 blue red),

(data 5 blue red 6.9 "avto"),

но не будет сопоставлен со следующими фактами:

(data 1.0 blue "red"),

(data 1 blue)

3) Одно- и многоместные переменные:

Одноместные: ?x, ?var, ?age Многоместные: $?y, $?zum

Представление правил в базе знаний. Типы условных элементов (1)

Пример:

```
(data 2 blue green),
(data 1 blue),
(data 1 blue red)
(defrule find-data-1
(data ?x ?y ?z)
=>
(printout t ?x " : " ?y " : " ?z crlf))
```

УЭ данного правила будет сопоставляться с первым и третьим фактом, поэтому в результате срабатывания правила будет выведено:

```
1 : blue : red
2 : blue : green
```

Представление правил в базе знаний. Типы условных элементов (1)

4) Ограничения со связками:

?x&red|blue трактуется как ?x&(red|blue), а не как (?x&red)|blue

```
(defrule r1
(data (value ?x&~red&~green))
=>
(printout t "slot value = " ?x crlf)).
```

Для факта (data (value blue)) это правило выведет сообщение:

slot value = blue

5) Предикатное ограничение задается с помощью символа “:”, за которым следует вызов предикатной функции. В качестве предикатных функций используются встроенные функции CLIPS: numberp, floatp, integerp, symbolp, stringp

Пусть заданы факты: ((data 1) (data 2) (data red)) Для определения значений числового типа: (data ?x&:(numberp ?x))

Представление правил в базе знаний. Типы условных элементов (1)

6) Ограничение возвращаемым значением:

Использует в качестве ограничения значение, возвращаемое внешней функцией. Эта функция вызывается непосредственно из УЭ-образца с использованием следующего синтаксиса:

=<вызов-функции>

Пример:

```
(defrule twice
(data (x  ?x) (y  =(* 2 ?x)))
=>...)
```

будет сопоставляться со всеми неупорядоченными фактами, у которых значение в слоте y равно удвоенному значению слота x

Представление правил в базе знаний. Типы условных элементов (2)

УЭ-проверка

(test <function-call>)

Удовлетворяется, если функция, вызываемая из него, возвращает значение отличное от FALSE.

В следующем правиле проверяется, что модуль разности двух чисел не меньше трех:

```
(defrule example-1
(data ?x)
(value ?y)
(test (>= (abs (- ?y ?x)) 3))
=>...)
```

Представление правил в базе знаний. Типы условных элементов (3)

Условный элемент “ИЛИ”

задается конструкцией (or <УЭ-1> ... < УЭ-N>)

Правило

```
(defrule r1
(man stud)
(or (spec computeer) (age 20))
=>...)
```

эквивалентно двум следующим:

```
(defrule r2
(man stud) (spec computeer)
=>...)
(defrule r3
(man stud) (age 20)
=>...)
```

Представление правил в базе знаний. Типы условных элементов (3)

Условный элемент “И”

задается (and <УЭ-1> ... < УЭ-N>)

Можно комбинировать УЭ “И” и “ИЛИ” в любых сочетаниях.

<u>Пример:</u>

```
(defrule r1
(sys-mode search)
(or (and (distance high) (resol little))
(and (distance low) (resol big)))
=>...)
```

Представление правил в базе знаний. Типы условных элементов (3)

Условный элемент “НЕ”

задается (not <УЭ>)

<u>Пример:</u>

```
(defrule not-double
(not (data red ?x ?x))
=>...)
```

Правило ищет факты, у которых второе поле – red, а третье и четвертое поля не совпадают.

Представление правил в базе знаний. Типы условных элементов (4)

Условный элемент “Существует”

синтаксис (exists <УЭ-1> ... <УЭ-N>)

Используется для определения удовлетворяется ли группа УЭ, специфицированных внутри условного элемента “Существует”, хотя бы одним набором образцов-сущностей в базе данных.

<u>Пример:</u>

правило:

```
(defrule example
(exists (a ?x) (b ?x))
=>...)
```

будет активизировано, если в базе данных имеется хотя бы одна пара фактов, содержащих в первых полях значения a и b, а вторые поля которых совпадают.

Представление правил в базе знаний. Типы условных элементов (5)

Условный элемент “Для всех”

синтаксис (forall <УЭ-1> ... <УЭ-N>)

Используется для определения, удовлетворяется ли группа УЭ, специфицированных внутри условного элемента “Для всех”, для каждого появления УЭ-1.

Пример: правило

```
(defrule all-students-passed
(forall  (student ?name)
(reading ?name)
(writing ?name)
(arithmetic ?name))
=> (printout t "All students passed." crlf))
```

активизируется, если каждый студент научился чтению, письму и арифметике.

Представление правил в базе знаний. Типы условных элементов (6)

Логические условные элементы

синтаксис (logical <УЭ>+)

Логический УЭ группирует образцы точно так же, как УЭ “И” и может использоваться в сочетании с УЭ “И”, УЭ “ИЛИ” и УЭ “НЕ”.

Однако логические УЭ можно применять только в первых образцах правила.

Представление правил в базе знаний. Типы условных элементов (6)

<u>Пример:</u> допустимо правило

```
(defrule ok
(logical (a))
(logical (b))
(c)
=>
(assert (d)))
```

но недопустимо правило

```
(defrule not-ok-1
(logical (a))
(b)
(logical (c))
=>
(assert (d)))
```

Логический вывод в системе CLIPS

Базовый цикл работы МЛВ в системе CLIPS:

1. Если достигнут предел активации правил или нет текущего фокуса – останов МЛВ. Иначе - выбор верхнего правила агенды модуля, которому принадлежит текущий фокус. Если в агенде нет правил, текущий фокус извлекается из стека фокусов и управление переходит к следующему модулю. Если стек фокусов пуст, то выполнение останавливается, в противном случае вновь выполняется шаг 1.
2. Выполняются операторы, содержащиеся в консеквенте выбранного правила. Счетчик числа правил инкрементируется.
3. Активированные правила помещаются в агенду модуля, в котором они определены. Деактивированные правила удаляются из агенды.
4. Значения значимостей всех правил, содержащихся в агенде переоцениваются. Цикл повторяется с шага 1.

Помещение правил в агенду

1. Вновь активируемые правила помещаются над всеми правилами с более низкой *значимостью* (salience) и ниже всех правил с более высокой значимостью.
2. Для определения места среди правил равной значимости используется текущая стратегия разрешения конфликта.
3. Если в результате добавления или удаления факта одновременно активизируются несколько правил и шаги 1 и 2 не позволяют выполнить упорядочение, то эти правила упорядочиваются между собой произвольно.

Назначение приоритетов правил

(defrule r1
(declare (salience 500))
(fire test-1)
=>
(printout t "Rule r1 firing." crlf))

Значимость назначается:

- при определении правила,
- при активизации правила (динамическая),
- в каждом цикле выполнения (динамическая).

Команда *set-salience-evaluation* используется для 2-го и 3-го случаев.

Стратегии разрешения конфликтов

1. *“вглубь”* (depth), используется по умолчанию,
2. *“вширь”* (breadth),
3. *“простоты”* (simplicity),
4. *“сложности”* (complexity),
5. *LEX (lex)*,
6. *MEA* (mea)
7. *случайного выбора* (random).

Команда *(set-strategy <strategy>) или меню “Execution/Options”*

(set-strategy <strategy>),

где <strategy>::= depth|breadth|simplicity |complexity| lex|mea|random.

Стратегии “вглубь” и “вширь”

ВГЛУБЬ: вновь активируемые правила помещаются в агенду **над** всеми правилами такой же значимости.

Факт f-1 активирует правила rule-1 и rule-2,

Факт f-2 активирует правила rule-3 и rule-4.

Тогда если f-1 устанавливается раньше, чем f-2, то rule-3 и rule-4 окажутся в агенде выше правил rule-1 и rule-2.

ВШИРЬ: Вновь активируемые правила помещаются **ниже** всех правил с такой же значимостью.

Стратегии “простоты” и “сложности”

ПРОСТОТЫ: активируемые правила помещаются **над** всеми правилами с **равной или большей** специфичностью.

(defrule example

(item ?x ?y ?x)

(test (and(numberp ?x) (> ?x (+ 10 ?y)) (< ?x 100)))

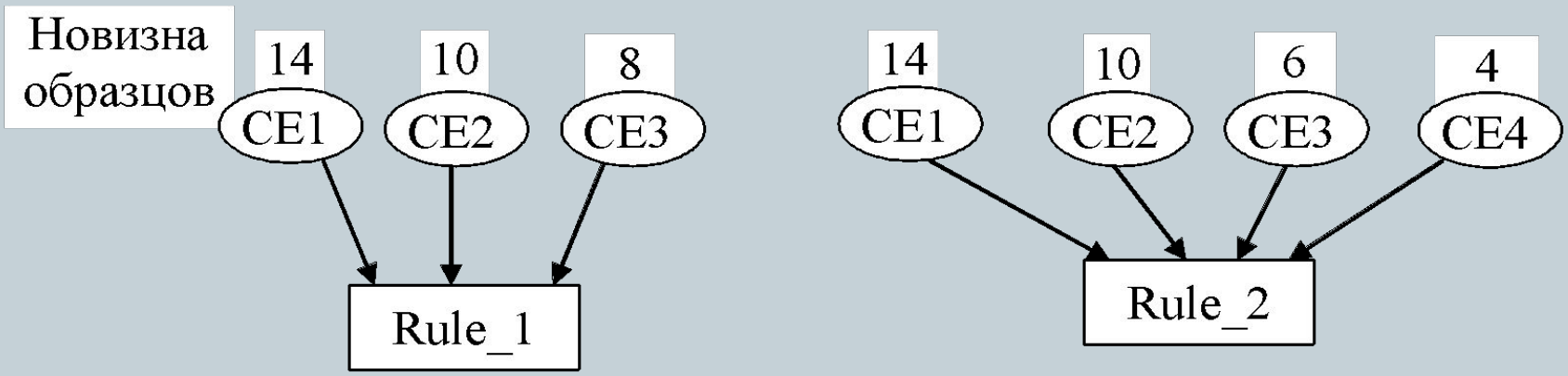
=>...)

имеет специфичность 5 (считаются операторы (item ?x ?y ?x), ?x, numberp, >, <).

СЛОЖНОСТИ: активируемые правила помещаются **над** всеми правилами с **равной или меньшей** специфичностью.

Стратегия LEX

Правило с **большим** значением **временного тега** помещается в агенду **выше** другого правила.

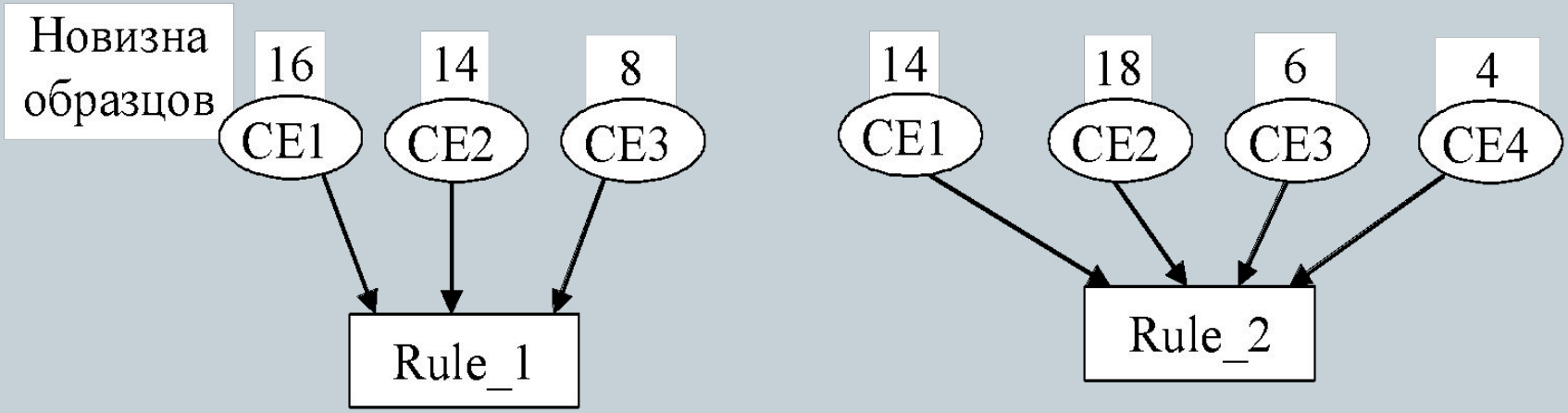


Сравнение правил в стратегии разрешения конфликтов LEX

Сработает правило Rule1, т.к. временной тег образца, связанного с его третьим условным элементом (8) больше, чем временной тег соответствующего образца у правила Rule2 (6).

Стратегия “MEA”

Правило с **большим** временным тегом **первого** условного элемента , помещается в агенду **выше**. Если временные теги первых образцов равны, то используется стратегия LEX.



Сравнение правил в стратегии разрешения конфликтов MEA

В примере на рисунке раньше сработает правило Rule1.

Стратегия случайного выбора

Каждому правилу сопоставляется случайное число, которое используется для определения его местоположения в агенде среди правил равной значимости.

Это случайное число сохраняется при изменении стратегии, а в случае возврата к случайной стратегии разрешения конфликта восстанавливается тот же порядок среди правил, которые находились в агенде, когда стратегия была изменена.

Фрагмент простой ЭС в среде CLIPS

```
(defrule data-input
 (initial-fact)
=>
 (printout t crlf "Введите число дней до зачета (целое значение): ")
 (bind ?days (read))
 (if (numberp ?days)
    then (assert (days ?days))
    else (printout t "Введите число" crlf))
  (printout t crlf "Введите число несданных лабораторных работ (в %)")
bind ?works (read))
  (assert (works ?works)))
(defrule R1
 (days ?days)
 (works ?works)
 (test (and (= ?days 1) (<> ?works 0)))
=>
(printout t crlf "Свободного времени нет" crlf)
(assert (freetime "no")))
```

Ссылка на материалы и рекомендуемая литература

Ссылка на файлы лекции, м/у к практическим работам и среды CLIPS в файловом хранилище mail.ru:

https://cloud.mail.ru/public/5L2A/5rzp8Y3ne

<u>Литература</u>

1. Пантелеев М.Г., Родионов С.В.
Модели и средства построения экспертных систем: Учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2003.– 68 с.

2. Частиков А.П., Гаврилова Т.А., Белов Д.Л.
Разработка экспертных систем. Среда CLIPS. СПб.: БХВ-Петербург, 2003. – 608 с.