СПбГЭТУ «ЛЭТИ»
Кафедра Вычислительной техники
Дисциплина «Искусственный интеллект»

Лекция 7
Логический вывод и стратегии разрешения конфликтов в системе CLIPS

© М.Г. Пантелеев, СПбГЭТУ «ЛЭТИ» 2021

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, на который указывает текущий фокус.*
   – Если агенда пуста, 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. Выполнение операторов, содержащихся в консеквенте выбранного правила. Инкремент счетчика числа правил.

3. Сопоставление. *Добавление активированных правил в агенду модуля*, в котором они определены. Удаление из агенды деактивированных правил

4. Переоценка значимостей всех правил, содержащихся в агенде. Переход на п. 1.

Добавление 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 значимости (приоритетов) правил

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

Значимость может назначаться:

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

Для динамического назначения приоритета правила используется команда:

`set-salience-evaluation`

Стратегии разрешения конфликтов в CLIPS

- *вглубь* (depth) // используется по умолчанию,
- *вширь”* (breadth)
- *простоты* (simplicity)
- *сложности* (complexity)
- *LEX (lex)*
- *MEA* (mea)
- *случайного выбора* (random).

Задаются с помощью команды **`(set-strategy <strategy>)`**

или меню “Ecexcution/Options”

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)))
=>…)
```

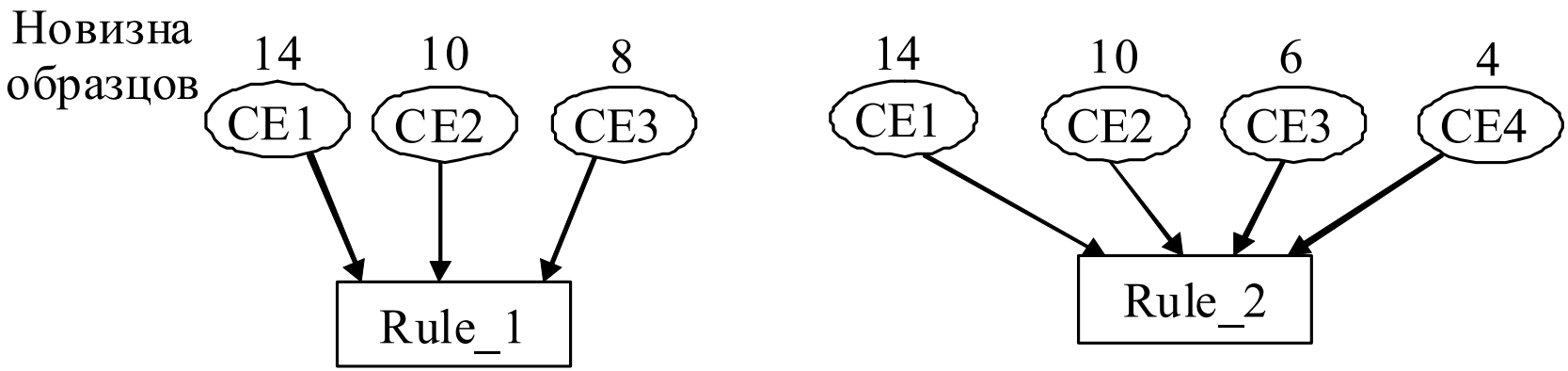
имеет специфичность 5 (считаются операторы (item ?x ?y ?x), ?x, numberp, >, <).

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

Стратегия LEX

Все факты помечаются временными тегами.

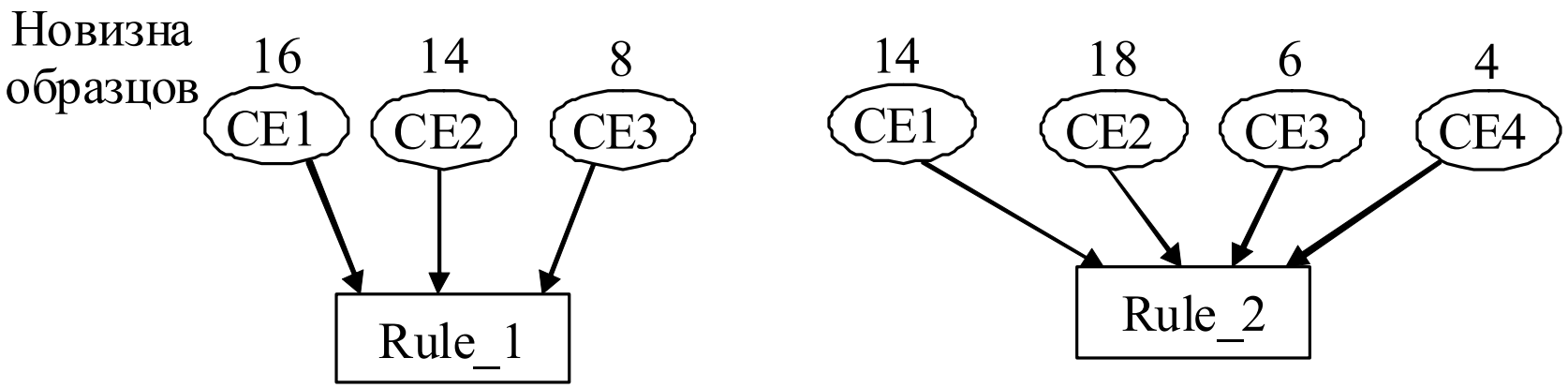
Правило с большим значением временного тега помещается в агенду выше другого правила



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



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) // всегда помещается в список фактов
=>
 (printout t crlf "Введите число дней до зачета (целое
  значение): ")
 (bind ?days (read)) // введенное значение присваивается переменной ?days
 (if (numberp ?days)
  then (assert (days ?days)) // добавление факта в список (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")))
```