Лекції 12.

CLIPS. Правила.

CLIPS підтримує евристичну й процедурну парадигму подання знань. Для подання знань у процедурній парадигмі CLIPS надає такі механізми, як глобальні змінні, функції й родові функції.
У цій лекції ми розглянемо такий спосіб подання знань, як ***правила***. Правила в CLIPS служать для подання евристик або так званих "емпіричних правил", які визначають набір дій, виконуваних при виникненні деякої ситуації. Розроблювач експертної системи визначає набір правил, які разом працюють над рішенням деякої задачі.

Правила складаються з ***передумов*** і ***нещастя***. Передумови називаються також **“ЯКЩО-частиною” *правила***, *лівою частиною правила* або *LHS правила* (left-hand side of rule).

Наслідок називається ***Т-частиною правила***, *правою частиною правила* або *RHS правила* (right-hand side of rule).

Передумови правила являють собою набір умов (або умовних елементів), які повинні задовольнитися, для того щоб правило виконалося.

Передумови правил задовольняються залежно від наявності або відсутності деяких заданих фактів у списку фактів або деяких створених об'єктів, що є екземплярами класів, визначених користувачем.

Один з найпоширеніших типів умовних виражень в CLIPS — ***зразки*** (patterns).

Зразки складаються з набору обмежень, які використаються для визначення того, чи задовольняє деякий факт або об'єкт умовному елементу. Інакше кажучи, зразок задає деяку маску для фактів або об'єктів.

Процес зіставлення зразків фактам або об'єктам називається ***процесом зіставлення зразків*** (pattern-matching).

CLIPS надає механізм, називаний ***механізмом логічного висновку*** (inference engine), що автоматично зіставляє зразки з поточним списком фактів і певних об'єктів у пошуках правил, які застосовні в цей момент.

Наслідок правила представляється набором деяких дій, які необхідно виконати, у випадку якщо правило застосовне до поточної ситуації.

Таким чином, дії, задані внаслідок правила, виконуються по команді механізму логічного висновку, якщо всі передумови правила задоволені.

У випадку якщо в цей момент застосовно більше одного правила, механізм логічного висновку використає так називану ***стратегію дозволу конфліктів*** (conflict resolution strategy), що визначає, яке саме правило буде виконано. Після цього CLIPS виконує дії, описані внаслідок обраного правила (які можуть вплинути на список застосовних правил), і приступає до вибору наступного правила. Цей процес триває доти, поки список застосовних правил не спорожніє.

Щоб краще зрозуміти сутність правил в CLIPS, їх можна представити у вигляді оператора **IF-THEN**, використовуваного в процедурних мовах програмування, наприклад, таких як Ada або C. Однак умови вираження **IF-THEN** у процедурних мовах обчислюються тільки тоді, коли потік керування програми безпосередньо попадає на даний вираз шляхом послідовного перебору виразів й операторів, що становлять програму.

В CLIPS, на відміну від цього, механізм логічного висновку створює й постійно модифікує список правил, умови яких у цей момент задоволені. Ці правила запускаються на виконання механізмом логічного висновку. Із цієї сторони правила схожі на оброблювачі повідомлень, що є присутнім у таких мовах програмування, як, наприклад, Ada або Smalltalk.

Без правил не обійдеться жодна експертна система, так що правила й мова їхнього подання в експертній системі можна сміло назвати найважливішою частиною будь-якої експертної оболонки. Успіх експертної системи багато в чому визначається тим, наскільки вдалий спосіб подання знань у вигляді правил, і наскільки добре їм володіє розроблювач експертної системи. Вся дана глава присвячена правилам, їхньому синтаксису й способам побудови, їхньому функціонуванню й призначенню, а також прийомам їхнього застосування.

9.1. Створення правил. Конструктор *defrule*

Для додавання нових правил у базу знань CLIPS надає спеціальний конструктор defrule. У загальному виді синтаксис даного конструктора можна представити в такий спосіб:

Визначення 9.1. Синтаксис конструктора defrule

```
(defrule
        <імені-правила>
        [<коментарі>]
        [<визначення-властивості-правила>]
        <передумови >                    ; ліва частина правила
        =>
        <наслідок>                       ; права частина правила
            )
```

Ім'я правила повинне бути значенням типу symbol. Як ім'я правила не можна використати зарезервовані слова CLIPS, які були перераховані раніше. Повторне визначення існуючого правила приводить до видалення правила з тим же ім'ям, навіть якщо нове визначення містить помилки.

Коментарі є необов'язковими, і, як правило, описують призначення правила. Коментарі необхідно містити в лапки. Ці коментарі зберігаються й надалі можуть бути доступні при перегладі визначення правила.

Визначення правила може містити оголошення властивостей правила, яких треба безпосередньо після імені правила й коментарів. Більш докладно властивості правила будуть розглянуті нижче.

У довідковій системі й документації по CLIPS для позначення передумов правила найчастіше застосовується термін "LHS of rule", а для позначення наслідку "RHS of rule", тому надалі ми будемо використовувати аналогічну термінологію - ліва й права частина правила.

Ліва частина правила задається набором умовних елементів, що звичайно складається з умов, застосованих до деяких зразків. Заданий набір зразків використовується системою для зіставлення з наявними фактами й об'єктами.

Всі умови в лівій частині правила поєднуються за допомогою неявного логічного оператора and. Права частина правила містить список дій, виконуваних при активізації правила механізмом логічного висновку.

Для поділу правої й лівої частини правил використовується символ =>. Правило не має обмежень на кількість умовних елементів або дій. Єдиним обмеженням є вільна пам'ять вашого комп'ютера. Дії правила виконуються послідовно, але тоді й тільки тоді, коли всі умовні елементи в лівій частині цього правила задоволені.

Якщо в лівій частині правила не зазначений жоден умовний елемент, CLIPS автоматично підставляє умову-зразок **initial-fact** або **initial-object**. Таким чином, правило активізується щораз із появою в

базі знань факту initial-fact АБО Об'єкта initial-object.

Якщо в правій частині правила не визначене жодне дія, правило може бути активоване й виконане, але при цьому нічого не відбудеться.

Більш докладно синтаксис лівої й правої частини правил буде описаний далі в цій лекції, а поки, як демонстрація застосування правил, напишемо найпростішу CLIPS-програму, що за традицією поздоровкається з усім світом, відразу після свого народження. Ви напевно знайомі з текстом подібних програм для процедурних мов програмування, таких як, наприклад, C. Мовою CLIPS така програма буде виглядати в такий спосіб:

Приклад 9.1. Програма "Hello-World!"

```
(clear)
(defrule
      Hello-World
      "My FirstCLIPS Rule"
      =>
      (printout  t  crlf crlf)
      (printout  t     ********************  crlf)
      (printout  t   "* HELLO WORLD!!!   *"  crlf)
      (printout  t     ********************  crlf)
      (printout  t  crlf  crlf)
      )
(reset)
(run)
```

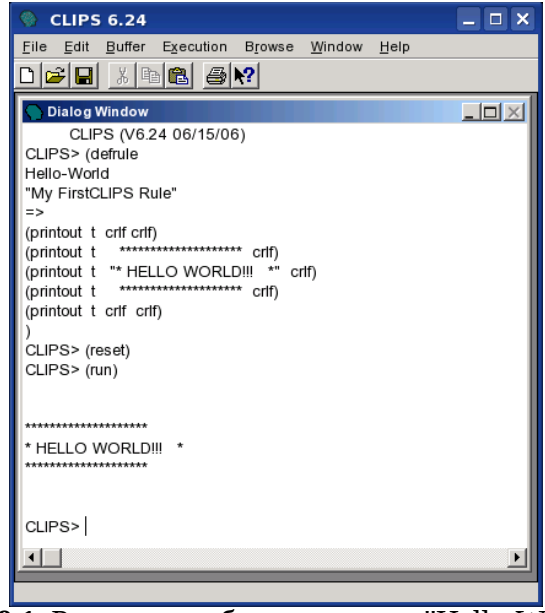

Рис. 9.1. Результат роботи програми "Hello-World!"

Тому що це перша наша програма мовою CLIPS, розберемо докладно всієї її дії й те, яким образом вони виконуються.

Функція **`clear`** повністю очищує систему, тобто видаляє всі правила, факти та інші об'єкти бази знань CLIPS, додані конструкторами, приводить систему в початковий стан, необхідний для кожної нової програми.

Потім, за допомогою конструктора defrule у систему додається нове правило з ім'ям Hello-World і відповідними коментарями. Ліва частина правила в цьому випадку відсутній, тому CLIPS автоматично формує передумови, що складаються з єдиного умовного вираження (initial-fact). Цей вираз є зразком найпростішого типу. При запуску програми на виконання механізм логічного висновку CLIPS буде шукати в списку фактів факт (initial-fact) і якщо він там буде знайдений — активізує правило. Права частина нашого правила складається з декількох викликів функції printout.

Зараз нам необхідно знати, що ця функція виводить текстовий вираз в один з потоків висновку. Параметр ***t*** задає стандартний потік висновку - екран. Він аналогічний, наприклад, стандартному потоку cout в C++. Вираження clrf служить для переходу на новий рядок.

Функція reset, як уже згадувалося раніше, очищає список фактів і заносить у нього факт (initial-fact), що дуже важливо для нормального функціонування нашої програми.

І, нарешті, функція run запускає механізм логічного висновку й приводить нашу програму в дію.

Якщо описані вище дії були виконані правильно, то ви повинні побачити результат, аналогічний наведеному на мал. 9.1 — фразу "HELLO WORLD!!!" у гарній рамочці із зірочок.

Щоб запустити нашу програму на виконання ще раз, досить викликати функції reset й run. Ці функції можна вводити із клавіатури, крім того, вони доступні в меню **Execution** і мають "гарячі" клавіші <Ctrl>+<E> й <Ctrl>+<R> відповідно.

CLIPS підтримує ряд функцій, команд і візуальних засобів, необхідних для ефективної роботи із правилами. Самі основні з них будуть розглянуті в даній лекції в міру необхідності.

Зараз же розглянемо візуальний інструмент, доступний користувачам Windows-версії середовища CLIPS — **Defrule Manager** (Менеджер правил). Для запуску менеджера правил у меню **Browse** виберіть пункт **Defrule Manager.** Зовнішній вигляд цього інструмента показаний на мал. 9.2.

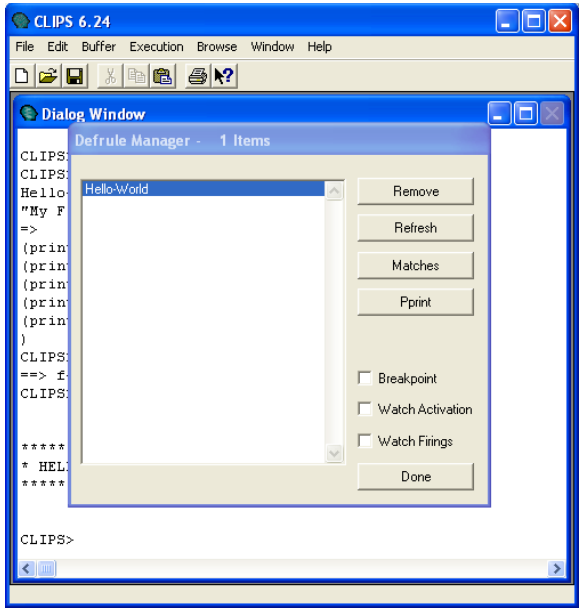

Рис. 9.2. Вікно менеджера правил

Менеджер відображає список правил, що є присутнім у системі в цей момент, і дозволяє виконувати над ними ряд операцій. Наприклад, за допомогою кнопки **Remove** можна видалити обране правило із системи, а за допомогою **Pprint** вивести у вікні CLIPS визначення виділеного правила разом з уведеними коментарями. Загальна кількість правил відображається в заголовку вікна менеджера — **Defrule Manager — 1 Items.**

Як ви вже могли переконатися, розбираючи наведений вище приклад нескладної програми, досить важко створити навіть мінімально корисну програму мовою CLIPS, не уявляючи собі, що саме відбувається при виконанні вашої програми. При створенні експертних систем необхідно точно знати, як відбувається зіставлення зразків, заданих у лівій частині правила, яким саме образом вибирається правило для виконання й т.д.

9.2. Основний цикл виконання правил

Після того як у систему додані всі необхідні правила й приготовлені початкові списки фактів й об'єктів, CLIPS готовий виконувати правила.

У традиційних мовах програмування крапка входу, крапка зупинки й послідовність обчислень явно визначаються програмістом.

В CLIPS потік виконання програми зовсім не вимагає явного визначення. **Знання** (правила) і **дані** (факти й об'єкти) розділені, і механізм логічного висновку, надаваний CLIPS, застосовує дані до знань, формуючи ***список застосовних правил***, після чого послідовно виконує їх. Цей процес називається ***основним циклом виконання правил*** (basic cycle of rule execution).

Розглянемо послідовність дій (кроків), виконуваних системою CLIPS у цьому циклі в момент виконання нашої програми:

1. Якщо були досягнуті межі виконання правил або не був установлений поточний фокус, виконання переривається. У противному випадку, для виконання вибирається перше правила модуля, на якому був установлений фокус. Якщо в поточному плані виконання немає вдоволених правил, то фокус переміщається по стеку фокусів і встановлюється на наступний модуль у списку. Якщо стік фокусів порожній, виконання припиняється. Інакше крок 1 виконується ще один раз.

2. Виконуються дії, описані в правій частині обраного правила. Використання функції return може міняти положення фокуса в стеці фокусів. Число запусків даного правила збільшується на одиницю, для визначення межі виконання правила.

3. У результаті виконання кроку 2 деякого правила можуть бути активовані або дезактивовані. Активовані правила (тобто правила, умови яких задовольняються в цей момент) містяться в план рішення задачі модуля, у якому вони визначені. Розміщення в плані визначається пріоритетом правила (salience) і поточною стратегією раз рішення конфліктів (ці поняття будуть описані нижче). Дезактивовані правила віддаляються з поточного плану рішення задачі. Якщо для правила встановлений режим перегляду активацій, то користувач одержить відповідне інформаційне повідомлення при кожній активації або дезактивації правила (режим перегляду активацій можна встановити за допомогою діалогового вікна **Watch Options.** Для цього виберіть пункт **Watch** у меню **Execution** й установите прапорець **Activations).**

4. Якщо встановлений режим динамічного пріоритету (dynamic salience), те для всіх правил з поточного плану рішення задачі обчислюються нові значення пріоритету. Після цього цикл повторюється із кроку 1.

9.3. Властивості правил

Для більше повного розуміння матеріалу, викладеного далі в цій лекції, необхідно розібратися з таким поняттям, як властивості правил. Властивості правил дозволяють задавати характеристики правил до опису лівої частини правила. Для завдання властивості правила використовується ключове слово declare. Одне правило може мати тільки одне визначення властивості, задане за допомогою declare.

Визначення 9.2. Синтаксис властивостей правил

<визначення-властивості-правила> ::= (declare <властивості-правила>)
<властивості-правила> ::= (salience <цілочисельне вираження>):
(auto-focus TRUE :FALSE)

9.3. 1. Властивість *salience*

Властивість правила salience дозволяє користувачеві призначати пріоритет для своїх правил. Пріоритет, що повідомляє, повинен бути виразом, що має цілочисельне значення з діапазону від -10 000 до +10 000.

Вираз, що представляє пріоритет правила, може використати глобальні змінні й функції . Однак намагайтеся не вказувати в цьому вираженні функцій, що мають побічну дію. У випадку якщо пріоритет правила явно не заданий, йому привласнюється значення за замовчуванням - 0.

Значення пріоритету може бути обчислене в одному із трьох випадків: при додаванні нового правила, при активації правила й на кожному кроці основного циклу виконання правил. Два останніх варіанти називаються ***динамічним пріоритетом*** (dynamic salience). За замовчуванням значення пріоритету обчислюється тільки під час додавання правила. Для зміни цієї установки можна використати команду set-salience-evaluation.

Крім того, користувачі Windows-версії середовища CLIPS можуть змінити це настроювання за допомогою діалогового вікна **Execution Options.** Для цього виберіть пункт **Options** у меню **Execution,** у діалоговому вікні, що з'явилося, укажіть необхідний режим обчислення пріоритету за допомогою списку, що **розкривається, Salience Evaluation**, як показано на мал. 9.3.

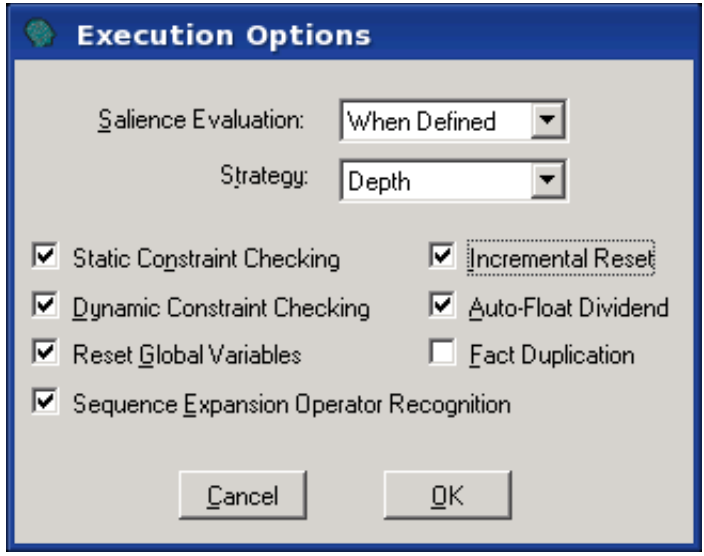

Рис. 9. 3. Установка способу обчислення пріоритетів правил

Зауваження

Кожний метод обчислення пріоритету містить у собі попередній (тобто якщо пріоритет обчислюється на кожному кроці основного циклу виконання правил, те він також обчислюється й при активації правила, а так само при його додаванні в систему).

9.3. 2. Властивість *auto-focus*

Властивість **auto-focus** дозволяє автоматично виконуватися команді focus при кожній активації правила.

Якщо властивість auto-focus установлена в значення TRUE, то команда focus у модулі, у якому визначене дане правило, автоматично виконується щораз при активації правила.

Якщо властивості auto-focus привласнене значення FALSE, то при активації правила не відбувається ніяких дій. За замовчуванням ця властивість установлена в FALSE.

6.4. Стратегія дозволу конфліктів

План рішення задачі — це список всіх правил, що мають задоволені умови при якомусь стані списку фактів й об'єктів (і які ще не були виконані).

Кожен модуль має свій власний план рішення задачі. Виконання плану подібно стеку (верхнє правило плану завжди буде виконано першим).

Коли активується нове правило, воно розміщається в плані рішення задачі керуючись наступними факторами:

1. Тільки що активоване правило міститься вище всіх правил з меншим пріоритетом і нижче всіх правил з більшим пріоритетом.

2. Серед правил з однаковим пріоритетом використовується поточна стратегія дозволу конфліктів для визначення розміщення серед інших правил з однаковим пріоритетом.

3. Якщо правило активоване разом з декількома іншими правилами, додаванням або виключенням деякого факту й за допомогою кроків 1 й 2 не можна визначити порядок правила в плані рішення задачі, то правило довільним образом упорядковуються разом з іншими правилами, які були активовані. Помітьте, що в цьому випадку порядок, у якому правила були додані в систему, робить довільний ефект на дозволи конфлікту (який найвищою мірою залежить від поточної реалізації правил). Намагайтеся не використати довільне упорядкування правил при рішенні задач, у яких потрібні точні результати або пояснення отриманих рішень.

CLIPS підтримує сім різних стратегій дозволу конфліктів: *стратегія глибини* (depth strategy), *стратегія ширини* (breadth strategy), *стратегія спрощення* (simplicity strategy), *стратегія ускладнення* (complexity strategy), *LEX* (LEX strategy), *MEA* (MEA strategy) і *випадкова стратегія* (random strategy).
За замовчуванням в CLIPS установлена стратегія глибини. Поточна стратегія може бути встановлена командою set-strategy (яка з поточний план рішення задачі, базуючись на новій стратегії). Крім того, користувачі Windows-версії середовища CLIPS можуть указати необхідну стратегію пошуку за допомогою діалогового вікна **Execution Options** (див. мал. 9.3). Для цього виберіть пункт **Options** у меню **Execution,** у діалоговому вікні, що з'явилося, виберіть необхідну стратегію за допомогою списку, що **розкривається, Strategy.**

6.4.1. Стратегія глибини

Тільки що активоване правило міститься вище всіх правил з таким же пріоритетом.

Наприклад, припустимо, що факт-а активував правило-1 і правило-2 і факт-б активував правило-3 і правило-4, тоді, якщо факт-а доданий перед фактом-б, у плані рішення задачі правило-3 і правило-4 будуть розташовуватися вище, ніж правило-1 і правило-2. Однак позиція правила-1 відносно правила-2 і правила-3 відносно правила-4 буде довільною.

6.4. 2. Стратегія широчини

Тільки що активоване правило міститься нижче всіх правил з таким же пріоритетом.

Наприклад, допустимо, що факт-а активував правило-1 і правило-2 і факт-б активував правило-3 і правило-4, тоді, якщо факт-а доданий перед в, у плані рішення задачі правило-1 і правило-2 будуть розташовуватися вище, ніж правило-3 і правило-4. Однак позиція правила-1 відносно правила-2 і правила-3 відносно правила-4 буде довільною.

6.4. 3. Стратегія спрощення

Між всіма правилами з однаковим пріоритетом тільки що активовані правила розміщаються вище всіх активованих правил з рівною або більшою ***визначеністю*** (specificity). Визначеність правила обчислюється за кількістю зіставлень, які потрібно зробити в лівій частині правила. Кожне зіставлення з константою або заздалегідь пов'язаної з фактом змінної додає до визначеності одиницю.

Кожний виклик функції в лівій частині правила, що є частиною умовних елементів :, = або test, також додає до визначеності одиницю. Логічні функції and, or й not не збільшують визначеність правила, але їхні аргументи можуть зробити це. Виклики функцій, зроблені усередині функцій, не збільшують визначеність правила.

Наприклад, випливаюче правило має визначеність, рівну 5.

Приклад 9.2. Обчислення визначеності правила

```
(defrule      example
         (item ?x  ?y  ?x)
         (test   (and   (numberp  ?x)    (>  ?x   (+
         10   ?y))    (<   ?x  100)))
         =>)
```

І порівняння заздалегідь зв'язаної змінної **?x** з константою, і виклики функцій numberp, < й > додають одиницю до визначеності правила. У підсумку одержуємо визначеність, рівну 5. Виклики функцій and й + не збільшують визначеність правила.

6.4.4. Стратегія ускладнення

Між правилами з однаковим пріоритетом, тільки що активовані правила розміщаються вище всіх активованих правил з рівною або меншою визначеністю.

6.4.5. Стратегія LEX

Між правилами з однаковим пріоритетом тільки що активовані правила розміщаються з використанням однойменної стратегії, уперше використаної в системі OPS5. Для визначення місця активованого правила в плані рішення задачі використовується "новизна" зразка, що активував правило. CLIPS маркірує кожен факт або об'єкт тимчасовим тегом для відображення відносної новизни кожного факту або об'єкта в системі.

Зразки, асоційовані з кожною активацією правила, сортуються по убуванню тегів для визначення місця розташування правила. Активація правила, виконана більше новими зразками, розташовується перед активацією, здійсненої більше пізніми зразками. Для визначення порядку розміщення двох активацій правил, поодинці рівняються відсортовані тимчасові теги для цих двох активацій, починаючи з найбільшого тимчасового тегу. Порівняння триває доти, поки не залишиться одна активація з найбільшим тимчасовим тегом. Ця активація розміщається вище всіх інших у плані рішення задачі.

Якщо активація деякого правила виконана більшим числом зразків, чим активація іншого правила й всі порівнювані тимчасові теги однакові, то активація з більшим числом тимчасових тегів поміщає перед активацією з меншим. Якщо дві активації мають однакову кількість тимчасових тегів й їхніх значень рівні, то правило з більшою визначеністю міститься перед активацією з меншої. На відміну від системи OPS5, умовний елемент not в CLIPS має псевдочасовий тег, що також використовується в даній стратегії дозволу конфліктів. Часовий тег умовного елемента not завжди менше, ніж часовий тег зразку.

Як приклад розглянемо наступні шість активацій правил, наведені в LEX-порядку (кома наприкінці рядка активації означає наявність логічного елемента not). Урахуйте, що тимчасові теги фактів не обов'язково дорівнюють індексу, але якщо індекс факту більше, те більше і його часовий тег. Для даного прикладу приймемо, що тимчасові теги дорівнюють індексам.

Приклад 9.3. Правила, відсортовані стратегією LEX

```
rule-6:                 f-1,f-4
rule-5:                 f-1,f-2,f-3,
rule-1:                 f-1,f-2,f-3
rule-2:                 f-3, f-1
rule-4:                 f-1,f-2
rule-3:                 f-2, f-1
```

У прикладі 9.4 показані ті ж активації з індексами фактів у тім порядку, у якому вони вирівнюються стратегією LEX.

Приклад 9.4. Порядок порівняння стратегією LEX

```
rule-6:   f-4,f-l
rule-5:   f-3,f-2,f-1,
rule-1:   f-3,f-2,f-1
rule-2:   f-3,f-1
rule-4:   f-2,f-1,
rule-3:   f-2,f-1
```

6.4.6. Стратегія MEA

Між правилами з однаковим пріоритетом тільки що активовані правила розміщаються *з* використанням однойменної стратегії, уперше використаної в системі OPS5.

Основна відмінність стратегії MEA від LEX полчгає у тому, що в стратегії MEA не виробляється сортування зразків, що активували правило. Рівняються тільки тимчасові теги перших зразків двох активацій. Активація з більшим тегом міститься в план рішення задачі перед активацією з меншим. Якщо обидві активації мають однакові тимчасові теги, асоційовані з першим зразком, то для визначення розміщення активації в плані рішення задачі використовується стратегія LEX. Так само, як й у стратегії LEX, умовний елемент not має псевдочасовий тег.

Як приклад розглянемо наступні шість активацій, наведені в Меа-порядке (кома на кінці активації означає наявність логічного елемента not).

Приклад 9.5. Правила, відсортовані стратегією MEA

```
rule-2:   f-3,f-l
rule-3:   f-2,f-l
rule-6:   f-l,f-4
rule-5:   f-1,f-2,f-3,
rule-1:   f-1,f-2,f-3
rule-4:   f-1, f-2,
```

6.4.7. Випадкова стратегія

Кожної активації призначається випадкове число, що використовується для визначення місця розташування серед активацій з однаковим пріоритетом. Це випадкове число зберігається при зміні стратегій, таким чином, той же порядок відтворюється при наступної установки випадкової стратегії (серед активацій у плані рішення задачі, коли стратегія замінена на вихідну).

6.5. Синтаксис LHS правила

Цей розділ описує синтаксис, використовуваний у лівій частині правил. Ліва частина правил містить список *умовних елементів* (conditional elements або CEs), які повинні задовольнятися, для того щоб правило було поміщено в план рішення задачі. Існує вісім типів умовних елементів, використовуваних у лівій частині правил: *CEs-зразки*, *test CEs*, *and CEs*, *or CEs*, *not CEs*, *exists CEs*, *forall CEs* й *logical CEs.*

Зразки - найбільше часто використовуваний умовний елемент. Він містить обмеження, які служать для визначення, чи задовольняє який-небудь елемент даних (факт або об'єкт) зразку.

Умова test використовується для оцінки вираження, як частини процесу зіставлення образів.

Умова and застосовується для визначення групи умов, кожне з якої повинне бути задоволене.

Умова or - для визначення однієї умови з деякої групи, що повинне бути задоволене.

Умова not - для визначення умови, що не повинне бути задоволене.

Умова exists - для перевірки наявності, принаймні одного, збігу факту (або об'єкта) з деяким заданим зразком.

І нарешті, умова logical дозволяє виконати додавання фактів і створення об'єктів у правій частині правила, пов'язаних з фактами й об'єктами, що збіглися із заданим зразком у лівій часта правила (підтримка вірогідності фактів у базі знань).

Синтаксис умовного елемента можна формалізувати в такий спосіб:

Визначення 9.3. Синтаксис умовного елемента

```
<умовн-елемент>   ::=   <pattern-CE> |
                        <assigned-pattern-CE> |
                        <not-CE> |
                        <and-CE> |
                        <or-CE> |
                        <logical-CE> |
                        <test-CE> |
                        <exists-CE> |
                        <forall-CE>
```

У наступній лекції (9) буде в тому числі докладно розглянуто синтаксис кожного умовного елемента.