Лекція 13.
CLIPS.
Правила (подовження).

9.5.1. Зразок (pattern РЄ)

Цей умовний елемент складається зі списку *обмежень полів, групових символів* (wildcards) і *змінних*, які використовуються для пошуку множини фактів або об'єктів, які відповідають заданому зразку. Таким чином, зразок як би визначає маску, який повинні відповідати дані. Такий умовний елемент задовольняється будь-яким фактом або об'єктом, що відповідають заданим обмеженням.

Обмеження полів — це набір обмежень, які використовуються для перевірки простих полів або слотів об'єктів. Обмеження полів можуть складатися тільки з одного символьного обмеження, однак, кілька обмежень можна з'єднувати разом. На додаток до символьних обмежень, CLIPS підтримує три інших типи обмежень: *об'єднуючі обмеження, предикатні обмеження* й *обмеження, що повертають значення*.

Групові символи використовуються при зіставленні зразків у ситуації, коли просте поле або група полів можуть приймати будь-які значення.

Змінні застосовуються для зберігання значення поля, що може бути згодом використане в лівій частині правила для іншого умовного елемента або в правій частині, як аргумент дії.

Перше поле будь-якого зразка обов'язково повинне бути значенням типу symbol і не може приймати значення інших типів. CLIPS використає перше поле для визначення: чи є даний зразок упорядкованим фактом, шаблоном або об'єктом. Ключове слово object зарезервоване для створення зразків, призначених для зіставлення з об'єктами. Будь-яке інше значення типу symbol повинне відповідати імені шаблона, створеного за допомогою конструктора deftemplate або неявно створеного шаблона. Для завдання імен слотів також повинні використовуватися значення типу symbol.

У слотах простих полів зразків, призначених для об'єктів і шаблонів, може втримуватися тільки одне обмеження поля, і не можуть бути присутнім групові символи або змінні. У складених слотах може втримуватися будь-яка кількість обмежень поля.
Далі будуть показані синтаксис і приклади використання зразків.

Для забезпечення наочності прикладів у наступних лекціях будуть використовуватися факти й шаблони, наведені в прикладі 9.6.

Приклад 9.6. Необхідні для подальшої роботи шаблони й факти

```
(deffacts                data-facts
          (data 1.0 blue "red")
          (data 1 blue)
          (data 1 blue red)
          (data 1 blue RED)
          (data 1 blue red 9.9))
(deftemplate          person
          (slot name)
          (slot age)
          (multislot friends))
(deffacts                 people
          (person (name Joe)  (age 20) )
          (person (name Bob)  (age 20) )
          (person (name Joe)  (age 34))
          (person (name Sue)  (age 34))
          (person (name Sue)  (age 20))
```

Символьні обмеження

Основні обмеження, що використовуються в зразках, — це обмеження, що визначають точну відповідність між полями факту й зразком. Ці обмеження називаються *символьними*. Символьне обмеження повністю складається з констант, таких як речовинні й цілі числа, значення типу symbol, рядки або імена об'єктів. Вони не можуть містити групових символів або змінних. Всі символьні обмеження при зіставленні зразків повинні точно збігатися по всіх зазначених полях, інакше факт не буде вважатися придатному даному зразку.

Умовний елемент, що представляє собою зразок для неупорядкованого факту, у якому присутні тільки символьні обмеження, має наступний синтаксис:

Визначення 9.4. Синтаксис символьних обмежень для неупорядкованого факту

(<обмеження-1> ... <обмеження-n>)

Умовний елемент, що представляє собою зразок для шаблона, у якому присутні тільки символьні обмеження, виглядає так:

Визначення 9.5. Синтаксис символьних обмежень для шаблона

(<ім'я-шаблону > (<ім'я-слота-1> <обмеження-1>)
…
(<ім'я-слота-n> <обмеження-n>))

Розглянемо приклад правил, що використають як зразок — зразок фактів (як упорядкованих, так і шаблонів) із символьними обмеженнями. Для нормальної роботи цього приклада необхідно ввести в CLIPS всі конструктори визначених фактів і шаблонів, представлені в цій лекції.
Після цього варто виконати команду reset для ініціалізації списку фактів. Для перевірки правильності виконаних операцій відкрийте вікно **Facts**. Якщо всі описані дії були виконані без помилок, ви повинні побачити результат, наведений на рис. 9.4.

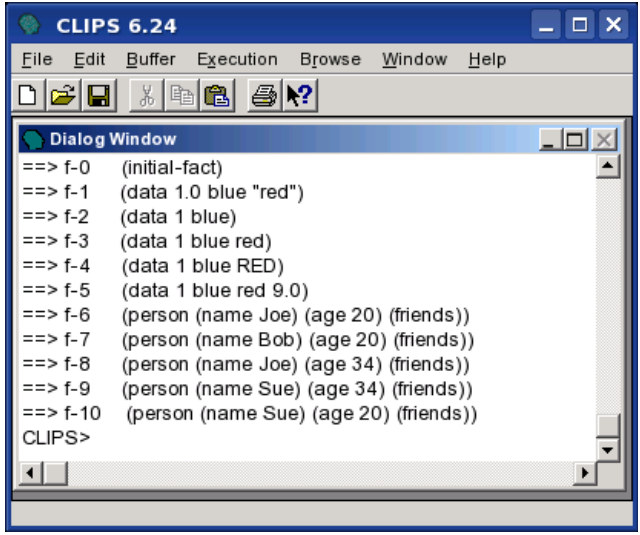

Рис. 9.4. Список необхідних фактів

Для нормальної роботи прикладів не забувайте виконувати команду reset перед кожним запуском правил. Уведіть в CLIPS визначення наступних правил:

Приклад 9.7. Правила із символьними обмеженнями

```
(defrule  Find-data
     (data  1  blue  red)
     =>
     (printout t crlf "Found data (data 1 blue red)" crlf))
(defrule Find-Bob-20
     (person (name Bob) (age 20))
     =>
     (printout t crlf "Found Bob-20 (person (name Bob) (age
     20))" crlf))
(defrule Find-Bob-30
     (person (name Bob) (age 30))
     =>
     (printout t crlf "Found Bob-30 (person (name Bob) (age
     30))" crlf))
```

Виконайте команди reset й run. Ви повинні одержати результат, наведений на мал. 9.5.

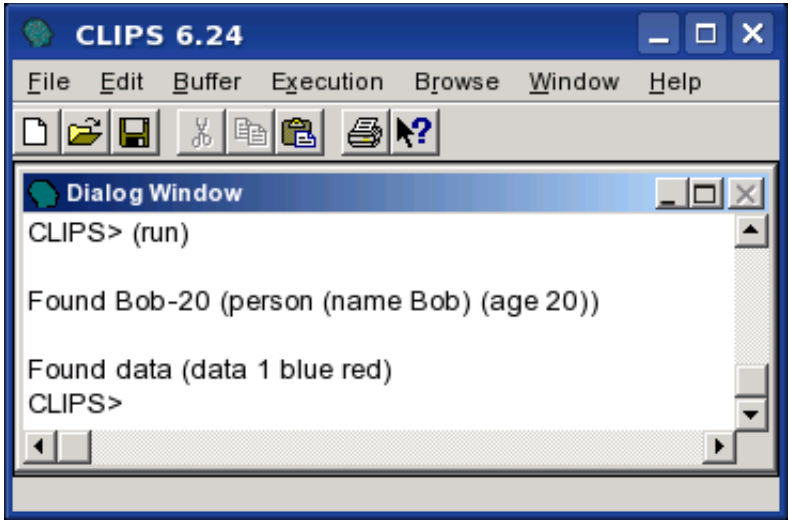

Рис. 9.5. Виконання правил із символьними обмеженнями

Як ми бачимо, були активовані й виконані два правила: Find-data й Find-Bob-20. Це відбулося тому, що зразки, задані в лівій частині цих правил, знайшли в списку фактів дані, повністю відповідаючи заданим символьним обмеженням.

Групові символи для простих і складових полів

В CLIPS є два різних групових символи, які використовуються для зіставлення полів у зразках. CLIPS інтерпретує ці групові символи як місце для підстановки деяких частин даних, що задовольняють зразкам. Груповий символ для простого поля записується за допомогою знака ?, що відповідає одному будь-якому значенню, збереженому в заданому полі. Груповий символ складеного поля записується за допомогою знака $? і відповідає, можливо, порожньої послідовності полів, збереженої в складеному полі. Групові символи для простих і складових полів можуть комбінуватися в будь-якій послідовності. Не можна використати груповий символ складеного поля для простих полів. За замовчуванням не заданий у зразку простій слот шаблона або об'єкта зіставляється з неявно заданим груповим символом для простого поля. Аналогічно не заданий у зразку складений слот зіставляється з неявно заданим груповим символом для складеного поля.

Умовний елемент, що представляє собою зразок для неупорядкованого факту, у якому присутні тільки символьні обмеження й групові символи, буде мати такий вигляд:

Визначення 9.6. Синтаксис обмежень для неупорядкованого факту

```
(<обмеження-1> ... <обмеження-n>)
<обмеження>      ::= <символьн-обмеження > : ? : $?
```

Відповідно для шаблона зразок прийме вид:

Визначення 9.7. Синтаксис обмежень для шаблона

```
(<ім'я шаблону >    (<ім'я-слота-1> <обмеження-1>)
                    …
                    (<ім'я-слота-n> <обмеження-n>))
```

Як приклад можна привести наступне правило:

Приклад 9.8. Правило Find-data

```
(defrule         Find-data
              (data   ? blue  red $?)   =>)
```

У нашому списку фактів присутні два факти, що підходять заданому шаблону й здатні активувати дане правило:

Приклад 9.9. Факти, що активують правило Find-data

```
(data  1  blue  red)
(data  1  blue  red  9.9))
```

Розглянемо ще одне правило:

Приклад 9.10. Правило match-all-persons

```
(defrule       match-all-persons
               (person)
               =>)
```

Оскільки person є шаблоном, а в зразку даного правила не визначений жоден слот шаблона, CLIPS автоматично поставить у відповідність кожному простому слоту груповий символ для простого поля, а складеному слоту - символ для складеного. Таким чином, правило перетвориться в наступне:

Приклад 9.11. Перетворене правило match-all-persons

```
(defrule      match-all-persons
               (person
               (name  ?)
               (age   ?)
               (friends  $?))
               =>)
```

Це правило буде активувати всі факти шаблона person.

Групові символи для складеного поля можна комбінувати із символьними обмеженнями, що приводить до одержання могутніших можливостей зіставлення зразків. Зразок, що зіставляється з усіма фактами, що мають значення YELLOW у будь-якому полі (включаючи перший), може бути записаний так:

Приклад 9.12. Зразок зі значенням YELLOW у будь-якому полі

```
(data $? YELLOW $?)
```

От кілька фактів, що відповідають цьому зразку:

Приклад 9.13. Факти зі значенням yellow у будь-якому полі

```
(data     YELLOW  blue  red  green)
(data     YELLOW  red)
(data     red YELLOW)
(data     YELLOW)
(data     YELLOW  data  YELLOW)
```

Останній факт буде відповідати зразку двічі, тому що yellow є присутнім у ньому двічі. Використання групового символу для складеного поля дозволяє створювати набагато більше загальні зразки, чим ті, які можна сформувати за допомогою групових символів для простого поля. Однак подібна спільність приводить до того, що процес зіставлення зразків, що використають групові символи, іноді займає набагато більше часу, чим аналогічний процес зі зразками, що використають тільки групові символи для простих полів.

Змінні, пов'язані із простими й складеними полями

Групові символи заміняють будь-які поля зразка й можуть приймати які завгодно значення цих полів. Значення поля може бути пов'язане зі змінними для наступного зіставлення, відображення й інших дій. Це виконується за допомогою застосування імені змінної наступної безпосередньо після групового символу.

Таким чином, синтаксис обмеження, застосовуваного в зразку, прийме наступний вид:

Визначення 9.8. Синтаксис обмежень

```
<обмеження> ::=  <символьн-обмеження > |
                  ? :
                  $? :
                  <змінна-простого-поля>|
                  <змінна-складеного-поля>
<змінна-простого-поля>       ::=  ?<ім'я-змінної>
<змінна-складеного-поля>   ::=  $?<ім'я-змінної>
```

Ім'я змінної повинне бути значенням типу symbol й обов'язково починатися з букви. В імені змінної не дозволяється використати лапки, тобто рядок не може використовуватися як ім'я змінної або її частина.

Правила зіставлення зразків при використанні змінних в обмеженнях зразка аналогічні правилам, що використовуються для групових символів. У момент першої появи імені змінної вона поводиться так само, як і відповідний груповий символ. У цей момент CLIPS зв'язує значення поля із заданої змінної. Цей зв'язок буде діяти тільки в рамках правила, у якому вона виникла. Кожне правило має свій власний список імен змінних зі значеннями, пов'язаними з ними, ці змінні локальні для правил.

Зв'язані змінні можуть бути використані в зовнішніх функціях. Символ $ має особливе значення в лівій частині правил - цей оператор відображає, що деяка, можливо порожня, послідовність полів вимагає зіставлення. У правій частині правила символ $ ставиться перед змінною для позначення того, що перед використанням змінної як аргумент функції необхідно розкрити послідовність полів, що втримуються в змінній. Таким чином, при використанні змінних як параметри функцій (як у лівої, так і правої частини правил) перед ім'ям змінне, утримуюче значення складеного поля, не повинен стояти символ $ (за винятком випадків, коли потрібно розкрити послідовність полів). При використанні змінне, утримуюче значення складеного поля, в інших випадках, перед її ім'ям повинен стояти символ $. Не можна застосовувати змінну складеного поля при операціях із простим полем зразка шаблона або об'єкта.

Як приклад уведіть у середовище CLIPS наступне правило:

Приклад 9.14. Правило Find-data

```
(defrule        Find-data
(data  ?  blue  ?x  $?y)   =>
(printout  t "Found  data  (data  ?  blue " ?x " " ?y ")"
crlf))
```

Виконайте команди reset й run. Якщо правило було уведено в систему без помилок, то на екрані з'явиться наступний результат:

Приклад 9.15. Результат роботи правила Find-data

```
Found  data    (data ?  blue  red  (6.9))
Found  data    (data ?  blue  RED  ())
Found  data    (data ?  blue  red  ()}
Found  data    (data ?  blue  red  ()}
```

Зразку, заданому в правилі, задовольняють чотири факти з індексами 1, 3, 4, 5. У результаті активації правило виводить на екран властивості фактів, що активували правило.

Розглянемо наступне правило:

Приклад 9.16. Модифіковане правило Find-data

```
(defrule   Find-data
         (data ?x $?y ?z) =>
         (printout  t   "x="   ?x  "  y="   ?y
                        "   z="   ?z  crlf))
```

Заданому зразку задовольняють всі факти data, але зверніть увагу, яким образом зв'язуються значення зі змінної в у різних випадках:

Приклад 9.17. Результат роботи модифікованого правила Find-data

```
x=1.0     y=(blue)       z=red
x=1       в=()           z=blue
x=1       y=(blue)       z=red
x=1       y=(blue)       z=RED
x=1       y=(blue red)   z=6.9
```

Після того як відбулося зв'язування змінної зі значенням, всі посилання на цю змінну повертають значення, з яким змінна була зв'язана. Це дійсно як для змінних, пов'язаних зі складеними полями, так і для змінних, пов'язаних із простими полями. Крім того, припустимі посилання між зразками в одному правилі.

Приклад 9.18. Правило Find-2-Coeval-Person

```
(defrule Find-2-Coeval-Person
          (person (name ?x) (age ?z))
          (person (name ?y) (age &z))
          =>
          (printout  t "name=" ?x " name=" ?y " age=" ?z crlf))
```

Наведене вище правило Find-2-Coevai-person виведе на екран усілякі пари імен людей (всі перестановки) однакового віку. Як навчити це правило не виводити еквівалентні за змістом або безглузді пари однакових імен (Bob-Bob), ми побачимо далі.

Єднальні обмеження

CLIPS надає 3 єднальні обмеження, призначених для об'єднання окремих обмежень і змінних у єдине ціле: & (логічне И), | (логічне АБО) і ~ (логічне НЕ). Обмеження & задовольняється, якщо два сусідніх обмеження задовольняються. Обмеження | задовольняється, якщо кожне із двох сусідніх обмежень задовольняється. Обмеження ~ задовольняється, якщо наступне за ним обмеження не задовольняється. Єднальні обмеження можуть комбінуватися майже довільним образом й у будь-якій кількості. Обмеження ~ має найвищий пріоритет, далі випливають & й |. У випадку однакового пріоритету обмеження обчислюється ліворуч праворуч. Існує одне виключення із правил пріоритету, що застосовується при зв'язуванні змінних. Якщо перше обмеження - це змінна й за нею треба &, то змінна є окремим обмеженням. Обмеження ?x&red|blue обчислюється як ?x& (red| blue), у той час як за правилами пріоритету воно повинне було обчислюватися як (?x&red) | blue.

Зв'язані обмеження мають наступний синтаксис:

Визначення 9.9. Синтаксис єднальних обмежень

```
<елемент-1>& <елемент-2> ... & елемент -n>
<елемент-1>| <елемент-2> ... | елемент -n>
~ <елемент>
```

Тут <елемент> повинен бути змінною, пов'язаною із простим або складовим полем, обмеженням або зв'язаним обмеженням.

Таким чином, визначення обмежень, наведені в попередніх розділах, можна розширити так:

Визначення 9.10. Синтаксис обмежень

```
<обмеження>     ::=  ?   :
                     $? :
                     <зв'язане-обмеження>
<зв'язане-обмеження>          ::=  <просте-обмеження>   |
     <просте-обмеження>&<зв'язане-обмеження>|
     <просте-обмеження>|<зв'язане-обмеження>|
<просте-обмеження>  ::=  <елемент>| ~  <елемент>
<елемент>       ::=  <константа>|
                     <проста-змінна>|
                     <складова-змінна>
```

Обмеження & звичайно служить тільки для об'єднання з іншими обмеженнями або зв'язування змінних. Помітьте, що єднальні обмеження можуть використати зв'язані змінні й у той же час самі робити зв'язування змінної зі значенням якогось поля. Якщо ім'я змінної зустрілося в перший раз, то для обмеження будуть використовуватися інші члени умовного елемента, а змінна буде пов'язана з відповідним значенням поля. Якщо змінна вже була зв'язана, то її значення працює як додаткове обмеження для даного поля.

Як приклад приведемо поліпшений варіант правила Find-2-coevai-Person з попереднього розділу.

Приклад 9.19. Поліпшене правило Find-2-Coeval-Person

```
(defrule    Find-2-Coeval-Person
            (person   (name  ?x)    (age  ?z))
            (person   (name  ?y&~?x)    (age  &z))
            =>
            (printout  t   "name="   ?x  "  name="   ?y  "  age="   ?z
            crlf))
```

Обмеження ?y&~?x забороняє виводити безглузді пари однакових імен (Bob-Bob). Однак дане правило усе ще виводить еквівалентні за змістом пари імен (наприклад, Bob-Sue й Sue-Bob).

Предикатні обмеження

Іноді необхідно обмежити поле, ґрунтуючись на істинності деякого логічного вираження. CLIPS дозволяє використати предикатні обмеження. Предикатні обмеження дозволяють викликати предикатні функції (функції, які повертають значення FALSE при не відповідності умовам і не-FALSE, якщо значення задовольняє умовам) протягом процесу зіставлення зразків. Якщо предикатна функція повертає значення НЕ-FALSE, обмеження задовольняється. Якщо предикатна функція повертає значення FALSE, то обмеження не задовольняється. Предикатні обмеження записуються за допомогою двокрапки й наступного за ним виклику відповідної предикатної функції. Звичайно предикатні обмеження використовуються спільно з єднальними обмеженнями й при зв'язуванні змінних (тобто якщо ви маєте змінну, котру потрібно зв'язати з деяким полем і хочете одночасно неї протестувати, об'єднаєте її із предикатним обмеженням).

Предикатні обмеження мають наступний синтаксис:

Визначення 9.11. Синтаксис предикатного обмеження

```
:<виклик-функції>
```

Таким чином, визначення поняття "елемент", наведене в попередньому розділі, можна розширити в такий спосіб:

Визначення 9.12. Синтаксис поняття "елемент"

```
<елемент>       ::= <константа>|
                    <проста-змінна>|
                    <складова-змінна>|
                    :<викликів-функції>
```

CLIPS надає кілька готових предикатних функцій. Крім цього, користувач також може створювати свої власні предикатні функції.

Приклад 9.20. Ще один варіант правила Find-data

```
(defrule   Find-data
           (data ?x&: (floatp ?x)&:{> ?x 0) $?y ?z&:(stringp ?z) )
           =>
           (printout t "x=" ?x " y=" ?y " z=" ?z crlf ) )
```

Вище наведений ще один варіант правила Find-data. У цьому випадку шукається факт неявно створеного шаблона data, перше поле якого - речовинне число більше нуля, а останнє - рядок. У нашому списку фактів такому правилу задовольняє тільки факт із індексом 1 -

```
(data  1.0 blue   "red") .
```

Обмеження, що повертають значення

В обмеженнях можливе використання значень, повернутих деякими функціями (у тому числі й зовнішніми). Виклик функції записується за допомогою знака = і зазначеної за ним функцією.

> **Зауваження**
> Функція порівняння також використає знак =. Різниця між ними може бути визначена по контексту.

Значення, що повертає, повинне бути одним із простих типів даних CLIPS. Це значення, повернуте функцією, поєднується зі зразком так, ніби воно було символьним обмеженням. Помітьте, що функція обчислюється при кожнім зіставленні зразків, а не один раз при визначенні правила.

Обмеження, що повертають значення, мають наступний синтаксис:

Визначення 9.13. Синтаксис обмеження, що повертає значення

=<виклик-функції>

Визначення поняття "елемент", наведені в попередньому розділі, приймуть такий вид:

Визначення 9.14. Синтаксис поняття "елемент"

```
<елемент>       ::= <константа>|
                    <проста-змінна>|
                    <складова-змінна>|
                    :<виклик-функції>
                    =<виклик-функції>
```

Правило із приклада 9.21 виводить на екран такі факти data, у яких значення другого поля у два рази більше, ніж значення першого. У нашому випадку це факти **(data 1 2)** та **(data 2 4)**.

Приклад 9.21. Використання обмеження, що повертає значення

```
(assert   (data 1 2)
          (data 2 3)
          (data 2 4))
(defrule  Find-data
          (data ?x ?y&=(* 2 ?x))
          =>
          (printout  t  "x="   ?x  "  y="   ?y crlf))
```