МЕТОДИ І СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

3 курс, весна 2021

- Доц. Баклан І.В.
- Email: iaa@ukr.net
- Web: baklaniv.at.ua

Лекція 15

Приклад розв`язання логічної задачі на CLIPS

Задача "Правдолюбці і брехуни"

Для того, щоб продемонструвати вам можливості мови CLIPS візьмемо логічну головоломку.

У головоломці вирішується одне із завдань, що виникають на острові, населеному мешканцями двох категорій: одні завжди говорять правду (назвемо їх правдолюбцем), а інші завжди брешуть (їх, природно, назвемо брехунами). Безліч подібних головоломок ви можете зустріти на сторінках цікавої книги Раймонда Смуляна «What is the Name of this Book?». Нижче наведені різні завдання з цієї серії.

Р1. Зустрічаються двоє людей, А і В, один з яких правдолюб, а інший - брехун. А говорить: «Або я брехун, або У правдолюб». Хто з цих двох правдолюб, а хто брехун?

Р2. Зустрічаються три людини, А, В і С. А і каже: «Всі ми брехуни», а В відповідає: «Тільки один з нас правдолюб». Хто з цих трьох правдолюб, а хто брехун?

РЗ. Зустрічаються три людини, А, В і С. Четвертий проходячи повз, запитує А: «Скільки правдолюбців серед вас?» А відповідає невизначено, а В відповідає: «А сказав, що серед нас є один правдолюб». Тут в розмову вступає С і додає: «В бреше!». Ким на вашу думку є В і С?

У програмі, вирішує проблеми подібного класу, будуть використані широкі можливості засобів програмування правил в мові CLIPS і продемонстровані деякі цікаві прийоми, наприклад використання контекстів і зворотного простежування. Ми також покажемо, як конструювати і тестувати прототипи, які приблизно відтворюють поведінку остаточної програми. Як зазначалося в основному матеріалі книги, технологія побудови експертних систем з використанням прототипів - одна з найпоширеніших в даний час.

Аналіз проблеми

Першим етапом будь-якого програмного проекту є аналіз розв'язуваної проблеми. Експерт повинен вміти вирішити проблему, а інженер по знаннях повинен розібратися, як саме було отримано рішення. При вирішенні нашого завдання вам доведеться виступити в обох іпостасях.

Запропоновані головоломки можна вирішити, систематично аналізуючи, що трапиться, якщо персонаж, що вимовляє репліку, є правдолюбцем, а що, якщо він - брехун. Позначимо через T (A) факт, що A каже правду, і отже, є правдолюбцем, а через F (A) - факт, що A бреше і, отже, є брехуном.

Розглянемо спочатку головоломку Р1. Припустимо, що А каже правду. Тоді з його репліки слід, що або А брехун, або У правдолюб. Формально це можна представити в наступному вигляді:

$$T(A) => F(A) \vee T(B)$$

Оскільки А не може одночасно бути і брехуном, і правдолюбцем, то це означає

$$T(A) => T(B)$$

Аналогічно можна записати і інший варіант. Припустимо, що А бреше:

$$F(A) => -(F(A) \vee T(B)).$$

Спростимо цей вислів:
F(A)=>-F(A) ^-T(B) или F(A)=> T(A)^F(B).

Порівнюючи обидва варіанти, неважко прийти до висновку, що толко останній правильний, оскільки в першому варіанті ми прийшли до висновку, що суперечить умовам (не можуть бути правдолюбцем одночасно A і B).

Таким чином, розглянута проблема відноситься до типу таких, вирішення яких перебуває в результаті аналізу висновків, що випливають з певних припущень, і пошуку в них протиріч (або відсутності таких). Ми припускаємо, що певний персонаж говорить правду, а потім дивимося, чи можна в цьому випадку так розподілити «ролі» інших персонажів, що не будуть порушені умови, сформульовані в репліках.

На жаргоні, прийнятому в математичне логіці, припущення про правдивість або брехливість безлічі висловлювань називається інтерпретацією, а варіант несуперечливого присвоєння значень істинності елементам безлічі - моделлю.

Однак наші головоломки включають і щось, що виходить за рамки типових проблем математичної логіки, оскільки репліки в них може вимовляти не один персонаж (як в головоломці P2), а на репліку одного персонажа може послідувати відповідь репліка іншого (як в головоломці P3). У початковій версії програми, яку ми розглянемо нижче, це ускладнення відсутня, але в остаточній воно повинно бути враховано. Ми покажемо, що поступове ускладнення програми досить добре узгоджується з використанням правил.

На практиці виявляється, що в першій версії програми найзручніше скористатися «виродженим» варіантом проблеми, тобто постаратися вирішити її в тривіальному вигляді, який, тим не менш, несе в собі багато особливостей реального випадку. Ось як це виглядає в відношенні наших правдолюбців і брехунів.

Р0. А заявляє: «Я брехун». Хто ж насправді А - брехун або правдолюб?

Ми тільки що фактично процитували добре відомий Парадокс Брехуна. Якщо А брехун, то, значить, він бреше, тобто в дійсності він правдолюб. Але тоді ми приходимо до протиріччя. Якщо ж А правдолюб, тобто говорить правду, то в дійсності він брехун, а це знову протиріччя. Таким чином, в цій головоломці не існує несуперечливого варіанта «розподілу ролей», тобто не існує моделі в тому сенсі, який надається їй в математичній логіці.

Є багато переваг у виборі для прототипу програми варіанти головоломки P0.

1. У головоломці присутній тільки один персонаж.

2. Вираз не містить логічних зв'язок, таких як І або АБО, або кванторів, на кшталт квантора спільності (все) і інших.

3. Відсутня відповідь репліка.

У той же час суттєві риси проблеми в цьому варіанті присутні. Ми як і раніше повинні спробувати відшукати несуперечливу інтерпретацію висловлювання А, тобто повинні реалізувати два завдання, присутні в будь-яких варіантах подібної головоломки:

- формувати альтернативні інтерпретації висловлювань;
- аналізувати наявність протиріч.

Онтологічний аналіз і представлення знань

Наступний етап - визначити, з якими видами даних нам доведеться мати справу при вирішенні цього класу головоломок. Які об'єкти представляють інтерес в світі правдолюбців і брехунів і якими атрибутами ці об'єкти характеризуються?

Мабуть, для вирішення завдань цього класу нам доведеться мати справу з такими об'єктами.

Персонажі, що вимовляють репліки. Усна репліка характеризує або самого персонажа, або інших персонажів, або й тих, і інших. Персонаж може бути або правдолюбцем, або брехуном.

Твердження, що міститься в репліці. Це твердження може бути або цілком брехливим, або абсолютно правдивим (істинним).

Трохи поміркувавши, ми прийдемо до висновку, що існують ще й інші об'єкти, які необхідно враховувати при вирішенні завдань цього класу.

Існує *середовище (світ)*, яка характеризується сукупністю наших припущень. Наприклад, існує *світ*, в якому ми припустили, що А - правдолюб, а отже, висловлене ним твердження (або затвердження) істинно. Це припущення тягне за собою різні наслідки, які утворюють контекст даного гіпотетичного світу.

Існує ще щось, що ми назвемо *причинами*, або *причинними зв'язками (reasons)*, які пов'язують висловлювання про той чи інший гіпотетичному світі. Якщо А стверджує, що «В - брехун», і ми припускаємо, що А - правдолюб, то це твердження є причиною (підставою), по якій ми можемо стверджувати, що в даному гіпотетичному світі У - брехун, а отже, всі твердження, які містяться в репліках, вимовних в, брехливі. Відстежуючи такі зв'язки між висловлюваннями, можна відновити початковий стан проблеми, якщо в результаті міркувань ми прийдемо до протиріччя.

Природно, що ці об'єкти можна представляти в програмі по-різному. Онтологічний аналіз практично ніколи не призводить до єдиного способу подання.

Для першої версії CLIPS-програми було вибрано таке уявлення описаних об'єктів:

```
;;  Об'єкт  statement  (вислів)  пов'язаний  з
певним
;; персонажем (поле speaker).
;;  Висловлювання  містить  твердження  (поле
claim).
;; Висловлювання має підставу - причину (поле
reason),
;; по якій йому можна довіряти,
;; і тег (tag) - це може бути довільний
;; ідентифікатор.
```

```
(deftemplate statement
            (field speaker (type SYMBOL))
                  (multifield claim (type
SYMBOL))
                (multifield reason (type
INTEGER) (default 0))
              (field tag (type INTEGER)
(default 1))
)
```

Замість того, щоб фокусувати внимение на персонажа, на чільне місце ставимо промовленою їм репліку (вислів), а персонаж відносимо до атрибутів висловлювання. Хочемо забезпечити можливість представити певну головоломку у вигляді примірника шаблону, наведеного нижче.

```
(statement (speaker A) (claim F A))
```

Цей шаблон можна перевести на «людську» мову наступним чином:

«Існує вислів, зроблене персонажем A, в якому стверджується, що A брехун і тег цього висловлювання за умовчанням отримує значення 1». Зверніть увагу на те, що в полі **`reason`** також буде встановлено значення за замовчуванням (це значення дорівнює 0), тобто ми можемо припустити, що ніяких попередніх висловлювань, які могли б підтвердити дане, в цьому завданні не було.

Зверніть увагу, що поля **`claim`** і **`reason`** мають кваліфікатор **`multifield`**, оскільки вони можуть містити кілька елементів даних.

Однак недостатньо тільки уявити в програмі висловлювання персонажів - нам знадобитися також виявити суть містяться в них тверджень. Далі, прийнявши певне припущення про правдивість або брехливість персонажа, якому належить вислів, можна побудувати гіпотезу про істинність або брехливості цього твердження. З кожним таким твердженням зв'яжемо унікальний числовий ідентифікатор.

```
;; Твердження, зміст якого, наприклад,
;; полягає в наступному,
;; T A. . . . означає, що A правдолюб;
;; F A. . . . означає, що A брехун.
;; Затвердження може мати під собою
;; підставу (reason) - зазвичай це тег
;; висловлювання (об'єкта statement) або тег
;; іншого твердження (об'єкта claim).
;; Затвердження також характеризується ознакою
scope,
;; який може приймати значення «істина» або
«брехня».
```

```
(deftemplate claim
    (multifield content (type SYMBOL))
      (multifield reason (type INTEGER)
(default 0))
            (field scope  (type SYMBOL))
)
```

Наприклад, розкривши вміст наведеного вище висловлювання в припущенні, що A каже правду, отримаємо наступні твердження (об'єкт **claim**):

```
(claim (content F A) (reason 1) (scope truth)).
```

Таким чином, об'єкт **claim** успадковує вміст від об'єкта **statement**. Останній стає обгрунтуванням (**reason**) цього твердження. Поле **scope** об'єкта **claim** приймає значення припущення про правдивість або брехливість цього висловлювання.

Ще нам буде потрібно уявлення в програмі того світу (world), в якому ми в даний час знаходимося. Об'єкти world породжуються в момент, коли ми формуємо певні припущення. Потрібно мати можливість розрізняти різні безлічі припущень і ідентифікувати їх в програмі в той момент, коли процес роздумів приводить нас до протиріччя. Наприклад, протиріччя між висловлюваннями T (A) і F (A) відсутня, якщо вони істинні в різних світах, тобто при різних припущеннях.

Світи будемо представляти у програмі наступним чином:

```
;; Об'єкт world являє контекст,
;; сформований певними припущеннями
;; про правдивість або брехливість персонажів.
;; Об'єкт має унікальний ідентифікатор в поле tag,
;;  а  сенс  допущення  -  істинність  або брехливість -
;; фіксується в поле scope.
```

```
(deftemplate world
             (field  tag  (type  INTEGER)
(default 1))
           (field  scope  (type  SYMBOL)
(default truth))
)
```

Зверніть увагу на те, що при зазначених в шаблоні значеннях за замовчуванням ми можемо починати кожен процес обчислень з об'єкта **`world`**, має в поле значення 1, причому цей «мир» можна заселити висловлюваннями персонажів, яких ми імовірно вважаємо правдолюбцем. Таким чином можна форматувати базу фактів **`the-facts`** для завдання P0 наступним чином:

```
;; Твердження, що A брехун.
(deffacts the-facts
          (world)
          (statement (speaker A)  (claim
F A))
)
```

Якщо цей оператор **deffacts** буде включений в той же файл, що і оголошення шаблонів (а також правила, про які йтиметься нижче), то після завантаження цього файлу в середу CLIPS нам знадобиться для запуску програми дати тільки команду **reset**.

Розробка правил

У цій частині лекції ми розглянемо набір правил, який допомагає впоратися з виродженим формулюванням P0 завдання про брехунів і правдолюба. Перші два правила, **`unwrap-true`** і **`unwrap-false`**, витягають вміст висловлювання на припущенні, що персонаж, якому належить вислів, є відповідно правдолюбцем або брехуном, і на цій підставі формують об'єкт **`claim`**.

```
        ;; Витяг вмісту висловлювання.
(defrule unwrap-true
         (world (tag ?N)  (scope truth))
         (statement (speaker ?X)  (claim $?Y)  (tag ?N))
=>
         (assert (claim (content T ?X) (reason ?N)
                        (scope truth)))
         (assert (claim (content $?Y) (reason ?M)
                        (scope truth)))  )
(defrule unwrap-false
         (world (tag ?N)  (scope falsity))
         (statement (speaker ?X)  (claim $?Y)  (tag ?N))
=>
         (assert (claim (content F  ?X)  (reason ?N)
                (scope falsity)))
```

```
(assert (claim (content NOT $?Y)  (reason ?N)
        (scope falsity)) )
```

У кожному з наведених правил перший оператор в умовній частині робить припущення відповідно про правдивість або брехливість персонажа, а другий оператор в заключній частині правила поширює припущення на формовані затвердження - об'єкти **`claim`**.

Далі нам знадобляться правила, які введуть заперечення в вирази. Оскільки T (A) еквівалентно F (A), а -F (A) еквівалентно T (A), то правила, які виконують відповідні перетворення, написати досить просто. Аналіз результатів застосування цих правил значно спростить виявлення суперечностей, що прямують з певного припущення.

```
;; Правила заперечення
(defrule not1
        ?F <- (claim (content NOT T ?P))
=>
        (modify ?F (content F ?P))
)
(defrule not2
        ?F <- (claim (conteny NOT F ?P))
=>
        (modify ?F (content T ?P))
)
```

```
;; Виявлення протиріччя між припущенням про
;; правдивості і наступними з нього фактами.
(defrule contra-truth
         (declare (salience 10))
         ?W <- (world (tag ?N) (scope truth))
         ?S  <- (statement (speaker ?Y) (tag ?N))
         ?P  <- (claim (content T ?X) (reason ?N) (scope
truth))
         ?Q <- (claim (content F ?X)  (reason ?N) (scope
truth))
=>
         (printout t crlf
                   "Statement is inconsistent if " ?Y "
is a knight."
```

```
;; "Висловлювання суперечливо, якщо"? Y "правдолюб."
     t crlf)
     (retract ?Q)
     (retract ?P)
     (modify ?W (scope falsity))
)
```

Якщо припустити, що оригінал висловлювання було правдивим, то в подальшому виявляється суперечлива пара тверджень, які потім видаляються з робочої пам'яті, а значення «правдивості» припущення в об'єкті **`world`** змінюється на **`falsity`** (брехливість). Якщо ж після цього також буде виявлено протиріччя, то ми приходимо до висновку про глобальну несумісності умов завдання, тобто в даній постановці ми маємо справу з логічним парадоксом.

```
;; Виявлення протиріччя між припущеннями про
;; брехливість і наступними з нього фактами.
(defrule contra-falsity
        (declare (salience 10))
        ?W <- (world (tag ?N) (scope falsity))
        ?S   <- (statement (speaker ?Y) (tag ?N))
        ?P   <- (claim (content F ?X) (reason ?N) (scope
falsity))
         ?Q  <- (claim (content T ?X) (reason ?N) (scope
falsity))
=>
 (printout t crlf
  "Statement is inconsistent if " ?Y " is a knave."
;; "Висловлювання суперечливо, якщо"? Y "брехун."
     t crlf)
     (modify ?W (scope contra))
)
```

Правило sweep забезпечує перевірку, чи всі наслідки з невірного припущення видалені з пам'яті.

```
;; Видалити з бази фактів всі твердження,
;; які випливають з припущення про правдивість.
(defrule sweep
      (declare (salience 20))
      (world (tag ?N) (scope falsity))
      ?F <- (claim (reason ?N) (scope truth))
=>
      (retract ?F)
)
```

Зверніть увагу на те, що правила **`contra-truth`**, **`contra-falsity`** і **`sweep`** мають більш високий пріоритет (значення параметра **`salience`**), ніж інші правила. Цим забезпечується якомога більш ранній виявлення протиріччя, а отже, і видалення з бази фактів тверджень, зроблених на основі припущення, що призвів до протиріччя.

Якщо тепер запустити на виконання програму, представивши їй початковий набір фактів, відповідних умові завдання P0, то програма знайде, що обидва контексту суперечливі. Іншими словами, незалежно від того, припускаємо ми, що A каже правду або бреше, програма виявить протиріччя в контексті **`world`**. Трасування програми в цьому випадку представлена в лістингу A.1. Рядки, виведені курсивом, - повідомлення основної програми, а інші - повідомлення програми трасування. Для зручності рядки, які вказують на активізацію правил, представлені напівжирним шрифтом.

Лістинг А.1. Трасування рішення задачі P0

```
CLIPS> (reset)
= => f-0  (initial-fact)
= => f-1  (world (tag 1) (scope truth))
= => f-2  (statement (speaker A) (claim F A) (reason 0)
(tag 1))
CLIPS> (run)
FIRE 1 unwrap-true: f-1,f-2
Assumption:
          A is a knight, so (T A) is true.
= => f-3  (claim (content F A) (reason 1) (scope truth))
= => f-4  (claim (content T A) (reason 1) (scope truth))
FIRE 2 contra-truth: f-1, f-2, f-4, f-3
Statement is inconsistent if A is a knight.
```

```
<= = f-3  (claim (content F A) (reason 1) (scope truth))
<= = f-4  (claim (content T A) (reason 1) (scope truth))
<= = f-1  (world (tag 1) (scope truth))
= => f-5  (world (tag 1) (scope falsity))
FIRE 3 unwrap-false: f-5, f-2
Assumption
A is a knave, so (T A) is false.
= => f-6   (claim (content NOT F A) (reason 1) (scope
falsity))
= => f-7    (claim (content F A) (reason 1) (scope
falsity))
FIRE 4 not2: f-6
<= = f-6   (claim (content NOT F A) (reason 1) (scope
falsity))
= => f-8    (claim (content T A) (reason 1) (scope
falsity))
FIRE 5 contra-falsity: f-5, f-2, f-7, f-8
```

```
Statement is inconsistent if A is a knave.
<= = f-5  (world (tag 1) (scope falsity))
= => f-9  (world (tag 1) (scope contra))
```

У наступній лекції ми ознайомимся з різноманітними прикладами застосування CLIPS.