Лекція 17
Приклад розв`язання логічної задачі на CLIPS

ЗАДАЧА “ПРАВДОЛЮБЦЫ И ЛЖЕЦЫ”

Для того, чтобы продемонстрировать вам возможности языка CLIPS возьмем логическую головоломку.

В головоломке решается одна из задач, возникающих на острове, населенном обитателями двух категорий: одни всегда говорят правду (назовем их правдолюбцами), а другие всегда лгут (их, естественно, назовем лжецами). Множество подобных головоломок вы можете встретить на страницах занимательной книги Раймонда Смуляна «What is the Name of this Book?». Ниже приведены разные задачи из этой серии.

Р1. Встречаются два человека, А и В , один из которых правдолюбец, а другой – лжец. А говорит: «Либо я лжец, либо В правдолюбец». Кто из этих двоих правдолюбец, а кто лжец?

Р2. Встречаются три человека, А, В и С. А и говорит: «Все мы лжецы», а В отвечает: «Только один из нас правдолюбец». Кто из этих троих правдолюбец, а кто лжец?

Р3. Встречаются три человека, А, В и С. Четвертый проходя мимо, спрашивает А: «Сколько правдолюбцев среди вас?» А отвечает неопределенно, а В отвечает: «А сказал, что среди нас есть один правдолюбец». Тут в разговор вступает С и добавляет: «В врет!» кем по-вашему являются В и С?

В программе, решающей проблемы подобного класса, будут использованы широкие возможности средств программирования правил в языке CLIPS и продемонстрированы некоторые интересные приемы, например использование контекстов и обратного прослеживания. Мы также покажем, как конструировать и тестировать прототипы, которые приблизительно воспроизводят поведение окончательной программы. Как отмечалось в основном материале книги, технология построения экспертных систем с использованием прототипов – одна из самых распространенных в настоящее время.

А.4.1. Анализ проблемы

Первым этапом любого программного проекта является анализ решаемой проблемы. Эксперт должен уметь решить проблему, а инженер по знаниям должен разобраться, как именно было получено решение. При решении нашей задачи вам придется выступить в обеих ипостасях.

Предложенные головоломки можно решить, систематически анализируя, что случится, если персонаж, произносящий реплику, является правдолюбцем, а что, если он – лжец. Обозначим через T(A) факт, что A говорит правду, и следовательно, является правдолюбцем, а через F(A) – факт, что A лжет и, следовательно, является лжецом.

Рассмотрим сначала головоломку P1. Предположим, что A говорит правду. Тогда из его реплики следует, что либо A лжец, либо

В правдолюбец. Формально это можно представить в следующем виде:

T(A) =>F(A) v T(B)

Поскольку А не может одновременно быть и лжецом, и правдолюбцем, то отсюда следует

T(A) =>T(B)

Аналогично можно записать и другой вариант. Предположим, что А лжет:

F(A)=>-(F(A) v T(B)).

Упростим это выражение:

F(A)=>-F(A) ^-T(B) или F(A)=> T(A)^F(B).

Сравнивая оба варианта, нетрудно прийти к выводу, что толко последний правильный, поскольку в первом варианте мы пришли к выводу, противоречащему условиям (не могут быть правдолюбцами одновременно А и В).

Таким образом, рассматриваемая проблема относится к типу таких, решение которых находится в результате анализа выводов, следующих из определенных предположений, и поиска в них противоречий (или отсутствия таковых). Мы предполагаем, что определенный персонаж говорит правду, а затем смотрим, можно ли в этом случае так распределить «роли» остальных персонажей, что не будут нарушены условия, сформулированные в репликах. На жаргоне, принятом в математическое логике, предположение о правдивости или лживости множества высказываний называется интерпретацией, а вариант непротиворечивого присвоения значений истинности элементам множества – моделью.

Однако наши головоломки включают и нечто, выходящее за рамки типовых проблем математической логики, поскольку реплики в них может произносить не один персонаж (как в головоломке Р2), а на реплику одного персонажа может последовать ответная реплика другого (как в головоломке Р3). В исходной версии программы, которую мы рассмотрим ниже, это усложнение отсутствует, но в окончательной оно должно быть учтено. Мы покажем, что постепенное усложнение программы довольно хорошо согласуется с использованием правил.

На практике оказывается, что в первой версии программы удобнее всего воспользоваться «вырожденным» вариантом проблемы, т.е. постараться решить ее в тривиальном виде, который, тем не менее, несет в себе многие особенности реального случая. Вот как это выглядит в отношении наших правдолюбцев и лжецов.

Р0. А заявляет: «Я лжец». Кто же в действительности А – лжец или правдолюбец?

Мы только что фактически процитировали хорошо известный *Парадокс Лгуна*. Если А лжец, то, значит, он врет, т.е. в действительности он правдолюбец. Но тогда мы приходим к противоречию. Если же А правдолюбец, т.е. говорит правду, то в действительности он лжец, а это опять противоречие. Таким образом, в этой головоломке не существует непротиворечивого варианта «распределения ролей», т.е. не существует модели в том смысле, который придается ей в математической логике.

Есть много достоинств в выборе для прототипа программы варианта головоломки P0.

- В головоломке присутствует только один персонаж.
- Выражение не содержит логических связок, таких как И или ИЛИ, или кванторов, вроде квантора общности (все) и прочих.
- Отсутствует ответная реплика.

В то же время существенные черты проблемы в этом варианте присутствуют. Мы по-прежнему должны попытаться отыскать непротиворечивую интерпретацию высказывания А, т.е. должны реализовать две задачи, присутствующие в любых вариантах подобной головоломки:

- формировать альтернативные интерпретации высказываниям;
- анализировать наличие противоречий.

A.4.2. Онтологический анализ и представление знаний

Следующий этап – определить, с какими видами данных нам придется иметь дело при решении этого класса головоломок. Какие объекты представляют интерес в мире правдолюбцев и лжецов и какими атрибутами эти объекты характеризуются?

По-видимому, для решения задач этого класса нам придется иметь дело со следующими объектами.

- *Персонажи*, произносящие реплики. Произносимая реплика характеризует либо самого персонажа, либо прочих персонажей, либо и тех, и других. Персонаж может быть либо правдолюбцем, либо лжецом.
- *Утверждение*, содержащееся в реплике. Это утверждение может быть либо целиком лживым, либо абсолютно правдивым (истинным).

Немного поразмыслив, мы придем к выводу, что существуют еще и другие объекты, которые необходимо учитывать при решении задач этого класса.

- существует *среда (мир)*, которая характеризуется совокупностью наших предположений. Например, существует *мир*, в котором мы предположили, что А – правдолюбец, а следовательно, высказанное им утверждение (или утверждения) истинно. Это предположение влечет за собой разные следствия, которые образуют контекст данного гипотетического мира.

- Существует еще нечто, что мы назовем *причинами*, или *причинными связями* (reasons), которые связывают высказывания о том или ином гипотетическом мире. Если А утверждает, что «В – лжец», и мы предполагаем, что А – правдолюбец, то это утверждение является причиной (основанием), по которой мы можем утверждать, что в данном гипотетическом мире В – лжец, а следовательно, все утверждения, которые содержатся в репликах, произносимых В, лживы. Отслеживая такие связи между высказываниями, можно восстановить исходное состояние проблемы, если в результате рассуждений мы придем к противоречию.

Естественно, что эти объекты можно представлять в программе по-разному. Онтологический анализ практически никогда не приводит к единственному способу представления.

Для первой версии CLIPS-программы было выбрано следующее представление описанных объектов:

```
;; Объект statement (высказывание) связан с определенным
;; персонажем (поле speaker).
;; Высказывание содержит утверждение (поле claim).
;; Высказывание имеет основание – причину (поле reason),
;; по которой ему можно доверять,
;; и тэг (tag) – это может быть произвольный
;; идентификатор.
(deftemplate statement
           (field speaker (type SYMBOL))
           (multifield claim (type SYMBOL))
                (multifield reason (type INTEGER)
(default 0))
        (field tag (type INTEGER) (default 1))
)
```

Вместо того, чтобы фокусировать внимение на персонаже, во главу угла ставим произносимую им реплику (высказывание), а персонаж относим к атрибутам высказывания. Хотим обеспечить возможность представить определенную головоломку в виде экземпляра шаблона, приведенного ниже.

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

Этот шаблон можно перевести на «человеческий» язык следующим образом:

«Существует высказывание, сделанное персонажем A, в котором утверждается, что A лжец и тэг этого высказывания по умолчанию получает значение 1». Обратите внимание на то, что в поле reason также будет установлено значение по умолчанию (это значение равно 0), т.е. мы можем предположить, что никаких предшествующих высказываний, которые могли бы подтвердить данное, в этой задаче не было.

Обратите внимание, что поля claim и reason имеют квалификатор multifield, поскольку они могут содержать несколько элементов данных.

Однако недостаточно только представить в программе высказывания персонажей – нам понадобиться также выявить суть содержащихся в них утверждений. Далее, приняв определенное предположение о правдивости или лживости персонажа, которому принадлежит высказывание, можно построить гипотезу об истинности или лживости этого утверждения. С каждым таким утверждением свяжем уникальный числовой идентификатор.

```
;; Утверждение, смысл которого, например,
;; состоит в следующем,
;; T A . . . .  означает, что A правдолюбец;
;; F A . . . . означает, что A лжец.
;; Утверждение может иметь под собой
;; основание (reason) – обычно это тэг
;; высказывания (объекта statement) или тэг
;; другого утверждения (объекта claim).
;; Утверждение также характеризуется признаком scope,
;; который может принимать значение «истина» или «ложь».
(deftemplate claim
    (multifield content (type SYMBOL))
    (multifield reason (type INTEGER) (default 0))
           (field scope  (type SYMBOL))
)
```

Например, раскрыв содержимое приведенного выше высказывания в предположении, что A говорит правду, получим следующие утверждение (объект claim):

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

Таким образом, объект claim наследует содержимое от объекта statement. Последний становится обоснованием (reason) данного утверждения. Поле scope объекта claim принимает значение предположения о правдивости или лживости этого высказывания.

Еще нам потребуется представление в программе того *мира* (world), в котором мы в настоящее время находимся. Объекты world порождаются в момент, когда мы формируем определенные предположения. Нужно иметь возможность различать разные множества предположений и идентифицировать их в программе в тот момент, когда процесс размышлений приводит нас к противоречию. Например, противоречие между высказываниями T(A) и F(A) отсутствует, если они истинны в разных мирах, т.е. при разных предположениях.

Миры будем представлять в программе следующим образом:

```
;; Объект world представляет контекст,
;; сформированный определенными предположениями
;; о правдивости или лживости персонажей.
;; Объект имеет уникальный идентификатор в поле tag,
;; а смысл допущения – истинность или лживость –
;; фиксируется в поле scope.
(deftemplate world
       (field tag (type INTEGER) (default 1))
       (field scope (type SYMBOL) (default truth))
)
```

Обратите внимание на то, что при указанных в шаблоне значениях по умолчанию мы можем начинать каждый процесс вычислений с объекта world, имеющего в поле значение 1, причем этот «мир» можно заселить высказываниями персонажей, которых мы предположительно считаем правдолюбцами. Таким образом можно инициализировать базу фактов the-facts для задачи P0 следующим образом:

```
;; Утверждение, что A лжец.
(deffacts the-facts
          (world)
          (statement (speaker A)  (claim F A))
)
```

Если этот оператор deffacts будет включен в тот же файл, что и объявления шаблонов (а также правила, о которых речь пойдет ниже), то после загрузки этого файла в среду CLIPS нам понадобится для запуска программы дать только команду reset.

A.4.3. Разработка правил

В этом разделе мы рассмотрим набор правил, который помогает справиться с вырожденной формулировкой P0 задачи о лжецах и правдолюбцах. Первые два правила, unwrap-true и unwrap-false, извлекают содержимое высказывания в предположении, что персонаж, которому принадлежит высказывание, является соответственно правдолюбцем или лжецом, и на этом основании формируют объект claim.

;; Извлечение содержимого высказывания.

```
(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))
)
```

В каждом из приведенных правил первый оператор в условной части делает предположение соответственно о правдивости или лживости персонажа, а второй оператор в заключительной части правила распространяет предположение на формируемые утверждения – объекты claim.

Далее нам понадобятся правила, которые введут отрицания в выражения. Поскольку –T(A) эквивалентно F(A),а –F(A) эквивалентно T(A), то правила, выполняющие соответствующие преобразования, написать довольно просто. Анализ результатов применения этих правил значительно упростит выявление противоречий, следующих из определенного предположения.

```
;; Правила отрицания
(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))
)
```

```
;; Выявление противоречия между предположением о
;; правдивости и следующими из него 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.”
```

```
;; “Высказывание противоречиво, если “ ?Y “ правдолюбец.”
     t crlf)
     (retract ?Q)
     (retract ?P)
     (modify ?W (scope falsity))
)
```

Если предположить, что исходное высказывание было правдивым, то в дальнейшем обнаруживается противоречивая пара утверждений, которые затем удаляются из рабочей памяти, а значение «правдивости» предположения в объекте world изменяется на falsity (лживость). Если же после этого также будет обнаружено противоречие, то мы приходим к выводу о глобальной несовместимости условий задачи, т.е. в данной постановке мы имеем дело с логическим парадоксом.

```
;; Выявление противоречия между предположениями о
;; лживости и следующими из него 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.”
;; “Высказывание противоречиво, если “ ?Y “ лжец.”
     t crlf)
     (modify ?W (scope contra))
)
```

Правило sweep обеспечивает проверку, все ли следствия из неверного предположения удалены из памяти.

```
;; Удалить из базы фактов все утверждения,
;; которые следуют из предположения о правдивости.
(defrule sweep
      (declare (salience 20))
      (world (tag ?N) (scope falsity))
      ?F <- (claim (reason ?N) (scope truth))
=>
      (retract ?F)
)
```

Обратите внимание на то, что правила contra-truth, contra-falsity и sweep имеют более высокий приоритет (значение параметра salience), чем другие правила. Этим обеспечивается как можно более ранее обнаружение противоречия, а следовательно, и удаление из базы фактов утверждений, сделанных на основе предположения, приведшего к противоречию.

Если теперь запустить на выполнение программу, представив ей исходный набор фактов, соответствующих условию задачи P0, то программа обнаружит, что оба контекста противоречивы. Другими словами, независимо от того, предполагаем ли мы, что A говорит правду или лжет, программа обнаружит противоречие в контексте world. Трассировка программы в этом случае представлена в листинге A.1. Строки, выведенные курсивом, - сообщения основной программы, а прочие – сообщения программы трассировки. Для удобства строки, указывающие на активизацию правил, представлены полужирным шрифтом.

Листинг А.1. Трассировка решения задачи 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))
```