Лекція 18
Приклад розв'язання логічної задачі на CLIPS. Варіанти

ЗАДАЧА “ПРАВДОЛЮБЦЫ И ЛЖЕЦЫ”

Нам предлагается рассмотреть другой вырожденный случай головоломок этого класса.

Предположим, что персонаж А утверждает : «Я всегда говорю правду». К какой категории следует отнести этот персонаж?

В такой постановке задача имеет неоднозначное решение. Предположение, что А правдолюбец, не приводит нас к противоречию. Но точно так же не приводит к противоречию и предположение, что А – лжец.

Наша задача – модифицировать описанную выше программу таким образом, чтобы она давала заключение, что оба контекста непротиворечивы. Один из возможных вариантов модификации – ввести в состав программы правила consist-truth и consist-falsity, разработав их по образцу правил contra-truth и contra-falsity. Эти правила должны дать пользователю знать, что при данном положении противоречий не обнаружено, причем правила должны активизироваться в случае, когда нет больше правил, претендующих на внимание интерпретатора.

Обратите внимание на то, что в задачах этого класса недостаточно убедиться, что начальное предположение об истинности некоторого высказывания не приводит к противоречию. Необходимо еще и проверить, приведет ли к противоречию обратное предположение. Если окажется, что оно также непротиворечиво, значит, задача не имеет единственного решения.

А.4.4. Расширение набора правил – работа с составными высказываниями

Расширим теперь возможности программы таким образом, чтобы она могла работать с составными высказываниями. Это даст возможность охватить в ней не только вырожденный случай, рассмотренный в предыдущем разделе, но и более сложные. За основу возьмем следующую головоломку.

P4. *Встречаются два персонажа, A и B, каждый из которых либо лжец, либо прадолюбец. Персонаж A говорит: «Мы оба лжецы.» К какой категории следует отнести каждого из них?*

В этой задаче нам придется иметь дело с конъюнкцией, поскольку утверждение, высказанное персонажем A, моделируется выражением

F (A) ^ F (B)

Эту конъюнкцию нужно разделить на выражения-компоненты и проанализировать их непротиворечивость. Очевидно, что A не может быть правдолюбцем, поскольку это противоречит утверждению, которое содержится в его реплике. Но программа должна самостоятельно «распаковать» эту конъюнкцию для того, чтобы прийти к такому выводу.

Нам также понадобится снабдить программу и средствами обработки дизъюнкции, поскольку, если предположить, что A лжет, нужно будет оперировать с отрицанием этого утверждения, которое преобразует выражение

F (A) ^ F (B)

в

F (A) v F (B)

Таким образом, в программу нужно включить правило выполнения отрицания составных высказываний и правило, которое «понимало» бы, что дизъюнкции вроде T (A) в действительности являются предположениями. Составное выражение T (A) v T (B) будем обрабатывать, предположив T (A), и проанализируем, нет ли в нем противоречия. Если таковое не обнаружится, то можно предположить, что T (A) v T (B) совместимо с утверждением о том, что A лгун, т.е. F (A). Но если предположение T (A) приведет к несовместимости, то нужно отказаться от него и предположить T (B). Если и это предположение приведет к несовместимости, то это озаначает, что утверждение T (A) v T (B) несовместимо с предположением F (A). В противном случае T (B) образует часть совместимоц интерпретации исходного высказывания.

В CLIPS составные высказывания проще всего представлять с помощью так называемой «польской» (или префиксной) нотации операций. Суть этого способа представления операций состоит в том, что символ операции предшествует символам операндов. Каждый оператор имеет фиксированное количество операндов, а потому всегда существует возможность однозначно установить область действия операций даже в случае, если операнды представляют собой вложенные выражения. Таким образом, выражение, представленное скобочной формой – (F (A)^T (B)), в польской записи будет иметь вид

NOT AND F A T B.

Легче всего восстановить исходный вид выражения, представленного в польской нотации, просматривая его справа налево. При этом операнды считываются до тех пор, пока не встретится объединяющий их оператор. Полученное выражение оказвается операндом следующего оператора. В представленном выше выражении B является операндом одноместного оператора T, а пара операндов T(B) и F(A) объединяется оператором AND.

Задавшись таким способом представления составных высказываний, сформируем правило выполнения отрицания дизъюнктивной и конъюнктивной форм, в котором будет использоваться функция flip, заменяющая “T” на “F” и наоборот.

```
(defrule not-or
       ?F <- (claim (content NOT OR ?P ?X ?Q ?Y))
=>
       (modify ?F (content AND (flip ?P)  ?X (flip ?Q)  ?Y))
)
(defrule not-and
        ?F <- (claim (content NOT AND ?P ?X ?Q ?Y))
 =>
        (modify ?F (content OR (flip ?P)  ?X  (flip ?Q)  ?Y))
)
```

Использование функции flip упрощает преобразование и позволяет перейти от выражения

NOT AND F A T B

Прямо к

OR T A F B,

Минуя

OR NOT F A NOT T B.

Функция flip определена следующим образом:

```
(deffunction flip (?P)
          (if (eq ?P T) then F else T)
)
```

Для упрощения мы ограничимся утверждениями в виде простых дизъюнкций или конъюнкций вида

T(A)vT(B) или F(A)^T(B),

Но не будем использовать более сложные утверждения в форме

F(B)^(T(A)vT(B))

Или

-(F(A)vF(B))^(T(A)vT(B)),

поскольку для решения большинства интересных головоломок вполне достаточно простых выражений.

Наибольшие сложности при модификации нашей программы связаны с обработкой дизъюнктивных выражений, поскольку вывод о наличии противоречия может быть сделан только после завершения анализа всех членов операндов дизъюнкции. Напрмер, нет противоречия между F(A) и T(A)vF(B). Противоречие, которое обнаружится при обработке первого операнда дизъюнкции T(A) в предположении F(A), будет локальным в контексте T(A). Но если мы вернемся к исходной дизъюнкции и попробуем проанализировать контекст F(B), то никакого противоречия обнаружено не будет, и, следовательно, интерпретация найдена.

Реализовать такой анализ локальных и глобальных противоречий можно, добавив в шаблон объекта claim атрибут contest:

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

Значение 0 в поле context означает, что мы имеем дело с глобальным контекстом, значение 1 – с локальным контекстом левого операнда, а значение2 – с локальным контекстом правого операнда дизъюнкции.

Пусть, например, анализируется дизъюнкция

T(A)vF(B),

Причем T(A) будет истинным в контексте 1, а F(B) – истинным в контексте 2. В этом случае все выражение будет истинным глобально, т.е. в контексте 0.

Структуру объекта world также нужно модифицировать – внести в нее поле context. Это позволит отслеживать ход вычислений. Пусть, например, объект world имеет вид

```
(world (tag 1)  (scope truth)  (context 2)).
```

Это означает, что данный «мир» создан следующей парой предположений:

- истинно высказывание, имеющее идентификатор (tag), равный 1, и
- правый операнд утверждения, которое содержится в этом высказывании, имеет значение «истина».

Новый вариант шаблона объекта world приведен ниже.

```
;; Объект world представляет контекст,
;; сформированный определенными предположениями
;; о правдтвости или лживости персонажей.
;; Объект имеет уникальный идентификатор в поле tag,
;; а смысл допущения – истинность или лживость –
;; фиксируется в поле scope.
;; В поле context сохраняется текущий контекст
;; анализируемого операнда дизъюнкции.
;; 0 означает глобальный контекст дизъюнкции,
;; 1 означает левый операнд,
;; 2 означает правый операнд.
(deftemplate world
        (field tag (type INTEGER)  (default 1))
        (field scope (type SYMBOL)  (default truth))
        (field context (type INTEGER)  (default 0))
)
```

Следующий шаг – разработка правил, манипулирующих контекстом. Приведенное ниже правило формирует контекст для левого операнда дизъюнкции.

```
(defrule left-or
        ?W <- (world (tag ?N)  (context 0))
        (claim (content OR ?P ?X ?Q ?Y)  (reason ?N)
                (scope ?V))
=>
        (modify ?W  (context 1))
        (assert  (claim
                 (content ?P ?X)  (reason ?N)  (scope ?V)
                             (context 1)))
)
```

Это правило устанавливает значение 1 в поле context объекта world т формирует соответствующий объект claim.

По этому же принципу разработаем правило для формирования контекста правого операнда дизъюнкции.

```
(defrule right-or
        ?W <- (world (tag ?N)  (context 1))
        (claim (content OR ?P ?X ?Q ?Y)  (reason ?N)
               (scope ?V))
=>
        (modify ?W (context 2))
        (assert  (claim
                (content ?Q ?Y)  (reason ?N)  (scope ?V)
                (context 2))
)
```

Далее разработаем правила, чувствительные к контексту, которые будут выявлять наличие противоречий в анализируемых утверждениях.

```
;; Выявление противоречия между предположением о
;; правдивости и следующими из него фактами
;; в разных контекстах одного и того же объекта world.
(defrule contra-truth-scope
        (declare (salience 10))
        (world (tag ?N) (scope truth) (context ?T))
        (claim
                (content T ?X) (reason ?N) (scope truth)
                (context ?S&: (< ?S ?T)))
        ?Q <- (claim (content P ?x) (reason ?N)
                (scope truth) (context ?T))
=>
        (printout t “Disjunct “  ?T
                “ is inconsistent with earlier truth context. “
;; “Дизъюнкт “ ?T
;; “ противоречит ранее установленному контексту правдивости. “
            crlf)
            (retract ?Q)
)
;; Выявление противоречия между предположением о
```

```
;; лживости и следующими из него фактами
;; в разных контекстах одного и того же объекта world.
(defrule contra-falsity-scope
        (declare (salience 10))
        ?W <- (world (tag ?N) (scope falsity) (context ?T))
        (claim
               (content F ?X) (reason ?N) (scope falsity)
               (context ?S&: (< ?S ?T)))
        ?Q <- (claim (content T ?X) (reason ?N)
               (scope falsity) (context ?T))
=>
        (printout t “Disjunct “ ?T
               “ is inconsistent with earlier falsity context. “
;; “Дизъюнкт” ?T
;; “ противоречит ранее установленному контексту лживости. “
               crlf)
   retract ?Q)
)
```

Нам потребуется модифицировать и прежний вариант правила contra-truth.

```
;; Выявление противоречия между предположением о
;; правдивости и следующими из него фактами
;; в одном и том же контексте оджного и того же объекта world.
(defrule contra-truth
       (declare (salience 10))
       ?W <- (world (tag ?N) (scope truth))
       ?P <-  (claim (content T ?X) (reason ?N) (context ?S)
                (scope truth))
       ?Q <- (claim (content F ?X) (reason ?N) (context ?S)
                (scope truth))
=>
       (printout t
                “Statement is inconsistent if “?X “ is a knight”
;; “Высказывание противоречиво, если “? X
;; “ правдолюбец.”
            crlf)
    (retract ?Q)
```

```
    (retract ?P)
    (modify ?W (scope falsity) (context 0)
)

;; Выявление противоречия между предположением о
;; лживости и следующими из него фактами
;; в одном и том же контексте одного и того же объекта world.
(defrule contra-falsity
       (declare (salience 10))
       ?W <- (world (tag ?N) (scope falsity))
       ?P <- (claim (content F ?X) (reason ?N) (context ?S)
             (scope falsity))
       ?Q <- (claim (content T ?X) (reason ?N) (context ?S)
             (scope falsity))
=>
       (printout t
             “Statement is inconsistent whether “ ?X
                    “ is a knight or knave.”
;; “Высказывание противоречиво, независимо от того,”
;; “является ли “ ?X “ прадолюбцем или лжецом.”
```

```
            crlf)
    (modify ?W (scope contra)
)
```

Поскольку теперь постановка задачи усложнилась по сравнению с вырожденным случаем, имеет смысл включить в программу распечатку предположений о характеристиках персонажей, упомянутых в высказываниях.

```
(defrule consist-truth
        (declare (salience -10))
         ?W <- (world (tag ?N) (scope truth)
        (statement (speaker ?Y) (tag ?N))
=>
        (printout t
        “Statement is consistent:”
;; “Высказывание непротиворечиво:”
     crlf)
     (modify ?W (scope consist)
)

(defrule consist-falsity
      (declare (salience -10))
```

```
      ?W <- (world (tag ?N) (scope falsity))
      (statement (speaker ?Y (tag ?N))
=>
      (printout t
            “Statement is consistent:”
;; “Высказывание непротиворечиво:”
            crlf)
      (modify ?W (scope consist)
)
(defrule true-knight
      (world (tag ?N) (scope consist))
      ?C <- (claim (content T ?X) reason ?N)
=>
      (printout t
      ?X “is a knight”
;;    ?X “ – правдолюбец”
              crlf)
          (retract ?C)
)
(defrule false-knave
```

```
        (world (tag ?M) (scope consist))
        ?C <- (claim (content F ?X) (reason ?N))
=>
        (printout t
               ?X " is a knave"
;;      ?X " – лжец"
                  crlf)
        (retract ?C)
)
```

Ниже приведенное правило разделения операции конюънкции, которое ранее мы предлагали вам разработать самостоятельно. Обратите внимание на то, что в нем также отслеживается контекст, хотя в данном случае без этого можно было бы и обойтись.

```
(defrule conj
       (world (tag ?N) (context ?S))
       (claim (content AND ?P ?X ?Q ?Y) (reason ?N)
                (scope ?V))
=>
       (assert (claim
                (content ?P ?X) (reason ?N) (scope ?V)
                (context ?S)))
       (assert  (claim
                (content ?Q ?Y) (reason ?N) (scope ?V)
                (context ?S)))
)
```

Прежде чем запустить программу на выполнение, сформируем исходные факты в соответствии с условиями задачи P4:

```
(deffacts the-facts
        (world)
        (statement (speaker A) (claim AND F A F B ) (tag 1))
)
```

После запуска программы в режиме трассировки интерпретатор сформирует распечатку процесса ее выполнения, приведенную в листинге А.2.

Листинг А.2. Трассировка решения задачи Р4

```
 CLIPS> (reset)
= => f-0  (initial-fact)
= => f-1  (world (tag 1) (scope truth) (context 0))
= => f-2  (statement (speaker A) (claim OR F A T B) (reason 0)
(tag 1))
CLIPS> (run)
FIRE 1 unwrap-true: f-1, f-2
Assumption
        F is a knight, so (OR F A T B) is true.
= => f-3  (claim (content OR F A T B) (reason 1) (scope truth)
(context 0))
= => f-4  (claim (content T A) (reason 1) (scope truth) (context
0))
FIRE 2 left-or: f-1, f-3
= => f-5  (claim (content F A) (reason 1) (scope truth) (context
1))
<= = f-1  (world (tag 1) (scope truth) (context 0))
= => f-6  (world (tag1) (scope truth) (context 1))
```

```
FIRE 3 contra-truth-scope: f-6, f-4, f-5
Disjunct 1 is inconsistent with earlier truth context.
<= = = f-5  (claim (content F A) (reason 1) (scope truth)
(context 1))
FIRE 4 right-or: f-6, f-3
= => f-7  (claim (content T B) (reason 1) (scope truth) (context
2))
<= = f-6  (world (tag 1) (scope truth) (context 1))
= => f-8  (world (tag 1) (scope truth) (context 2))
FIRE 5 consist-truth: f-8, f-2
Statement is consistent:
<= = f-8  (world (tag 1) (scope truth) (context 2))
= => f-9  (world (tag 1) (scope consist) (context 2))
FIRE 6 true-knight: f-9, f-7
B is a knight
<= = f-7  (claim (content T B) (reason 1) (scope truth) (context
2))
FIRE 7 true-knight: f-9, f-4
A is a knight
```

```
<= = f-4  (claim (content T A) (reason 1) (scope truth) (context
0))
CLIPS>
```

A.5. Обратное прослеживание и множество контекстов

Модифицируем программу таким образом, чтобы она могла справиться и с задачами этого класса в более сложной постановке. Речь идет о задачах, в которых несколько персонажей произносят реплики. Пример такого рода головоломки приведен ниже.

Упражнение 3

P5. *Встречаются два человека, A и B, которые заявляют следующее.*

A: «Я говорю правду, либо B лжец».

B: «A говорит правду, либо я лжец».

К какой категории следует отнести каждый из персонажей?

Задача анализа высказываний нескольких персонажей потребует использования более сложной методики, которая получила наименование «обратное прослеживание на основе анализа зависимостей» (dependency-directed backtracking).

От программы потребуется выполнить обратное прослеживание (откат) в следующих ситуациях:

- когда обнаружится конфликт между текущим «миром» и ранее существовавшим, причем в ранее существовавшем «мире» предполагается истинность высказывания, но не была проанализирована его лживость;
- когда обнаружится конфликт между текущим «миром» и ранее существовавшим, причем в ранее существовавшем «мире» был проанализирован только один операнд в составном дизъюнктивном утверждении.

Чтобы смысл этих формулировок стал более понятным, рассмотрим следующий пример.

P6. *Встречаются два человека, A и B, которые заявляют следующее.*

A: «Хотя бы один из нас говорит правду».

B: «Хотя бы один из нас лжец».

К какой категории следует отнести каждый из персонажей?

Высказывания персонажей представим в следующем виде:

A: T(A) v T(B)

B: F(A) v F(B)

Начнем с заявления персонажа B

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

И проанализируем левый операнд дизъюнкции. В результате будет сформирована корректная непротиворечивая интерпретация: B – правдолюбец, A – лжец.

Получив непротиворечивую интерпретацию высказывания персонажа B, перейдем к анализу высказывания персонажа A:

T(A)=> FALSE

Поскольку правдивость A противоречит сформированной ранее интерпретации высказывания персонажа B. Предположим, что A – лжец. Тогда:

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

Таким образом, оказывается, что это предположение также не работает, поскольку противоречит выбранной ранее интерпретации высказывания персонажа B, из которой следует, что B говорит правду.

Но анализ высказывания персонажа B нельзя считать законченным, поскольку не был выполнен анализ правого операнда дизъюнкции

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

И не было проанализировано предположение, что B лжец. До тех пор, пока это не будет выполнено, мы не имеем права делать вывод, что высказывания в формулировке задачи противоречат друг другу.

Поэтому придется вернуться назад в ту точку процесса логического анализа, где было сделано предположение об истинности левого операнда в дизъюнкции, и проанализировать вместо него правый операнд F(B). При этом сразу же будет обнаружено противоречие между истинностью F(B) и ранее высказанным предположением о правдивости персонажа B, но, не вернувшись назад и не выполнив этот анализ, мы не смогли бы обнаружить это противоречие. Теперь остается проанализировать следствие из предположения, что B – лжец.

F(B)=> -(F(A) v F(B))=> T(A) ^ T(B)=> FALSE

Только теперь можно с чистой совестью утверждать, что не существует непротиворечивой интерпретации высказываний, приведенных в условии задачи. Предположение о правдивости персонажа B приводит к конфликту с высказыванием персонажа A, а предположение о лживости B противоречит его же словам.

Чтобы в системе, использующей правила в качестве основного программного компонента, реализовать откат (обратное прослеживание), нужно в первую очередь иметь возможность восстановить тот контекст, который существовал в момент, когда было сформулировано предположение, приведшее к не удовлетворяющему нас результату. Одно из достоинств продукционных систем, подобных CLIPS, состоит в том, что они способны выполнить такой откат, не сохраняя прежнего состояния процесса вычислений, что коренным образом отличает их от фундаментально рекурсивных языков программирования, таких как LISP и PROLOG. При возникновении необходимости выполнить откат продукционные системы последовательно отменяют в обратном порядке все операции, связанные с добавлением данных в рабочую память, которые были выполнены, начиная с точки возврата, в которую нужно вернуться, вплоть до текущего этапа вычислений.

Но таким способом можно реализовать возврат, только предполагая, что в ходе выполнения операций, следующих за точкой возврата, из рабочей память не было удалено ничего существенного, а все действия, модифицирующие состояние рабочей памяти, носили исключительно аддитивный характер.

Примеры, подобные задаче Р6, существенно усложняют жизнь, поскольку для их решения программа должна выполнять некоторые дополнительные операции, в которых не было необходимости при решении задач с единственным высказыванием.

(1) Сохранять информацию о возможных точках возврата.

(2) При обнаружении противоречия принимать решение, выполнять или не выполнять откат, а если выполнять, то в какую именно точку.

(3) Отменить все изменения, внесенные в состояние рабочей памяти после «прохождения» выбранной точки возврата.

(4) Возобновить вычисления начиная с точки возврата.

Рассмотрим подробнее каждую из этих операций.

- Каждый объект world имеет уникальный числовой идентификатор, который хранится в поле tag. Эта информация практически не используется при решении задач с единственным высказыванием, поскольку мы всегда имеем дело с одним и тем же объектом world, связанным с этим высказыванием. Но при решении задач, оперирующих с несколькими высказываниями, нам придется различать утверждения, которые порождены разными высказываниями в разных «мирах». По мере того, как мы будем переходить от анализа одних высказываний к другим, будут формироваться и новые объекты world. Прежние объекты world нужно оставлять в таком состоянии, чтобы при необходимости к ним можно было еще раз вернуться. Это означает, что вектор world, с которым прекращены операции (возможно, временно), содержал всю

информацию, которая потребуется программе для возобновления работы с ним.

- При этом именно та точка, в которой процесс вычислений «переключился» на новый объект world, и будет, потенциальной точкой возврата. Информация, сохраняемая в объекте, включает знание о том, какое предположение о правдивости или лживости персонажа было сделано в этом «мире» и какие дизъюнкты (операнды составного дизъюнктивного выражения) в утверждении, содержащемся в высказывании персонажа, уже проанализированы.

- Поскольку каждый объект world имеет свой уникальный идентификатор и каждое утверждение (объект claim) индексировано определенным объектом world , можно довольно просто выяснить, существует ли противоречие между разными «мирами» (т.е. между утверждениями, связанными с разными объектами world). Остается единственный вопрос – нужно ли возвращаться в ранее покинутый «мир», если в текущем «мире» обнаружено противоречие с ним. Мы будем применять стратегию поиска в глубину, которая состоит в том, что откат нужно выполнять только в том случае, если противоречие сохраняется после полного завершения анализа текущего «мира».

- Если объекты world нумеруются последовательно, по мере их формирования, то потребуется разработать правило, которое при возвращении в покинутый ранее «мир» уничтожит как текущий объект world, так и все промежуточные объекты такого типа, которые при необходимости затем могут быть воссозданы.
- Если прежний объект world содержит полную информацию о том, в каком состоянии был покинут «мир», и утверждения в этом «мире» не противоречат этому состоянию, то ничто не мешает нам продолжить вычисления из точки возврата.

Начнем модификацию нашей программы с того, что в шаблон объекта world включим слот, в котором будет храниться идентификатор ранее покинутого «мира» (объекта), с которым данный объект конфликтует. Это нужно сделать по двум причинам.

(1) Нам потребуется различать случаи, в которых противоречия возникают в пределах одного и того же «мира», от конфликтов между «мирами». Если текущее высказывание само по себе противоречиво (т.е. является парадоксом), нет смысла выполнять откат в прежний мир и искать в нем разрешения противоречия.

(2) Наличие такого слота позволит разработать правило, которое будет выполнять откат прямо в этот покинутый ранее «мир».

Ниже будет показано, что для решения проблемы можно обойтись без реализации правила, упомянутого в п.2., хотя это и не так легко сделать, но соображения, высказанные в п.1., в любом случае остаются в силе.

```
;; Объект world представляет контекст,
;; сформированный определенными предположениями
;; о правдивости или лживости высказывания,
;; принадлежащего некоторому персонажу.
;; Объект имеет уникальный идентификатор в поле tag,
;; а смысл допущения – истинность или лживость –
;; фиксируется в поле scope.
;; Поле prior может содержать идентификатор
;; объекта world, обработанного перед тем,
;; как был создан данный объект, и с которым данный
;; объект может потенциально конфликтовать.
;; В поле context сохраняется текущий контекст
;; анализируемого операнда дизъюнкции.
(deftemplate world
```

```
      (field tag (type INTEGER) (default 1))
      (field scope (type SYMBOL) (default truth))
      (field prior (type INTEGER) (default 0))
      (field context (type INTEGER) (default 0)
)
```

Помимо модификации структуры объекта, для выполнения отката потребуется разработать правила для выполнения некоторых ключевых операций. Эти операции перечислены ниже вместе с ключевыми словами, ассоциированными с каждой из них.

- CHECK. Эта операция реализует нормальный режим выполнения вычислений при анализе предположений о правдивости или лживости.
- CONTRA. Анализ обнаруженного противоречия. Возникло ли оно между двумя высказываниями в одном и том же «мире»? Возникло ли противоречие между двумя высказываниями в одном и том же «мире», но в разных контекстах, например в разных операндах дизъюнкции? Возникло ли оно между двумя разными «мирами», производными от высказываний разных персонажей?

- CLEAN. После того, как выявлен характер возникшего противоречия, и перед тем, как выполнить откат в точку возврата, эта операция удаляет все утверждения, созданные в текущем «мире».
- BACK. Если мы имеем дело с противоречием между текущим «миром» и ранее покинутым, эта операция выполняет возврат в ранее покинутый «мир», в котором не был полностью завершен анализ всех дизъюнктов или не было проанализировано предположение о лживости.

- QUIT. Нам потребуется обнаружить ситуацию, которая наступает в случае, когда проанализированы все возможные интерпретации множества высказываний, т.е. все дизъюнктивные ветви и все возможные комбинации предположений о правдивости или лживости высказываний. Если при обнаружении такой ситуации не удалось найти непротиворечивую интерпретацию, можно со всей ответственностью утверждать, что условия задачи сами по себе несовместимы, т.е. не существует ее решения в терминах отнесения каждого из персонажей к определенной категории – к лжецам или правдолюбцам.

Еще раз модифицируем определение шаблона объекта world – внесем в него поле TASK, в котором будут представлены перечисленные задачи. Это поле будет использовано правилами, которые нам еще предстоит разработать. Механизм работы с задачами подобен тому, который использовался для манипулирования лексемами управления (control tokens), описанными в главах 5 и 14. Этот механизм активизирует определенные правила. Однако при этом мы не будем использовать стратегию MEA или специальные векторы. Лексемы управления будут просто сохраняться в определенном поле объекта world. Но результат будет тот же – эта лексема будет использована для активизации определенного правила.

```
;; Объект world представляет контекст,
;; сформированный определенными предположениями
;; о правдивости или лживости высказывания,
;; принадлежащего некоторому персонажу.
;; Объект имеет уникальный идентификатор
;; в поле tag, который соответствует
;; тегу высказывания.
;; Смысл допущения – истинность или лживость –
;; фиксируется в поле scope.
;; Поле TASK содержит одно из перечисленных
;; ниже значений:
;; CHECK – анализ предположений о
;;     правдивости или лживости высказывания;
;; CONTRA – анализ обнаруженного противоречия;
;; CLEAN – удаляет все утверждения, созданные
;;     в противоречивом мире ;
;; BACK – откат в точку возврата;
;; QUIT – прекращение процесса.
;; Поле prior может содержать идентификатор
;; объекта world, обработанного перед тем,
```

```
;; как был создан данный объект, и с которым данный
;; объект может потенциально конфликтовать.
;; В поле context сохраняется текущий контекст
;; анализируемого операнда дизъюнкции.
(deftemplate world
     (field tag (type INTEGER)  (default 1))
     (field scope (type SYMBOL)  (default truth))
     (field task (type SYMBOL)  (default check))
     (field prior (type INTEGER)  (default 0))
     (field context (type INTEGER)  (default 0))
)
```

Теперь разработаем правила, которые будут выполнять перечисленные выше операции. Кроме того, нужно внести некоторые изменения и в правила, разработанные для прежней версии программы.

Выявление противоречий

В процессе решения задач о правдолюбцах и лжецах могут быть обнаружены логические противоречия двух типов:

- между высказываниями в одном и том же «мире», но , возможно, в разных контекстах дизъюнктивного утверждения;
- между высказываниями в разных «мирах».

Для анализа вариантов возникновения противоречий целесообразно разработать четыре правила. Разобьем первую из указанных ситуаций на два случая:

- обнаруживается противоречие между предположением и высказыванием, которые существуют в одном и том же контексте (например, если из предположения T(A) непосредственно следует заключение F(A), как в заявлении персонажа A: F(A));
- обнаруживается противоречие между предположением и одним из дизъюнктов составного высказывания, как в заявлении персонажа A: T(B) v F(A).

Ситуацию, когда противоречие существует между высказываниями в разных “мирах”, тоже можно разделить на два случая:

- текущий “мир” рассматривается в предположении, что персонаж говорит правду (в поле scope объекта world содержится значение truth);
- текущий «мир» рассматривается в предположении, что персонаж лжет (в поле scope объекта world содержится значение falsity).

Если, предположив правдивость персонажа, программа обнаружит противоречие, она должна проанализировать и следствие из противоположного предположения – что персонаж лжец. И только при условии, что оба варианта предположения приводят к противоречию, нужно выполнить откат.

Анализ каждого из четырех вариантов ситуации выполняется отдельным правилом, программы которых представлены ниже. Обратите внимание, что все правила имеют довольно высокий приоритет (значение параметра salience). Это обеспечивает их первоочередную активизацию механизмом разрешения конфликтов между правилами. Кроме того, правила, анализирующие противоречие в пределах одного и того же «мира», имеют более высокий приоритет, чем правила, анализирующие противоречие между разными «мирами». Тем самым обеспечивается реализация стратегии по возможности избегать откатов в процессе решения проблемы.

```
;; ЕСЛИ обнаруживается противоречие между предположением
;; и производными от него фактами в пределах одного и
;; того же “мира” и в одном и том же контексте,
;; ТО зафиксировать противоречие и удалить
;; противоречивые утверждения (объекты claim)
;; из базы фактов.
(defrule contradiction
     (declare (salience 100))
     ?W <- (world (tag ?N) (task check) (context ?S)
          (prior 0))
     ?P <- (claim (content ?F ?X) (reason ?N)
          (context ?S))
     ?Q <- (claim (content ?G&: (not (eq ?G ?F)) ?X)
          (reason ?N (context ?S))
=>
     (printout
          t crlf
          “CONTRADICTION: “ ?F ?X “ versus “
          ?G ?X “in world “ ?N
;;   “ПРОТИВОРЕЧИЕ между: “ ?F ?X “ и “?G ?X “в мире” ?N
```

```
            t crlf)
      (retract ?P)
      (retract ?Q)
      (modify ?W (task contra))
)

;; ЕСЛИ обнаруживается противоречие между предположением
;; и производными от него фактами в пределах одного и
;; того же “мира”, но в разных контекстах,
;; ТО зафиксировать противоречие.
(defrule transcontext
      (declare (salience 90))
      ?W <- (world (tag ?N) (task check) (context ?T)
            (prior 0))
      (claim (content ?F ?X) (reason ?N)
            (context ?S&: (< ?S ?T)))
      (claim (content ?G&: (not eq ?G ?F)) ?x)
            (reason ?N) (context ?T))
=>
      (printout t crlf
```

```
          “TRANSCONTEXT CONTRADICTION: “ ?F ?X ” versus “
          ?G ?X “ in world “ ?N
;;    “ТРАНСКОНТЕКСТНОЕ ПРОТИВОРЕЧИЕ между: “ ?F ?X
;;    “ и “ ?G ?X “ в мире “ ?N
          t crlf)
     (modify ?W (task contra))
)

;; ЕСЛИ обнаруживается противоречие между
;; текущим “миром” в предположении о правдивости
;; и ранее покинутым “миром”,
;; ТО зафиксировать противоречие.
(defrule transworld-truth
     (declare (salience 80))
?W <- (world (tag ?N) (scope truth) (task check)
          (prior 0))
     (claim (content ?F ?X)  (reason ?N))
     (claim (content ?G&: (not (eq ?G ?F)) ?X)
     (reason ?M&: (< ?M ?N)))
=>
```

```
     (printout
          t crlf
          “TRANSWORLD CONTRADICTION: “ ?F ?X “ versus “
          ?G ?X “ in worlds “ ?N “I” ?M
;;   “МЕЖМИРОВОЕ ПРОТИВОРЕЧИЕ: “ ?F ?X “ противоречит “
;;   ?G ?X “ в мирах “ ?N “I” ?M
          t crlf)
     (modify ?w (task contra))
)

;; ЕСЛИ обнаружится противоречие между
;; текущим “ миром” в предположении о лживости
;; и ранее покинутым “миром”,
;; ТО подготовиться к выполнению отката в ранее
;; покинутый “мир”.
(defrule transworld-falsity
     (declare (salience 80))
     ?W <- (world (tag ?N) (scope falsity) (task check))
     (claim (content ?F ?X) (reason ?N))
     (claim (content ?G&: (not (eq ?G ?F)) ?X)
```

```
          (reason ?M&: (< ?M ?N)))
=>
     (printout
          t crlf
          “TRANSWORLD CONTRADICTION: “ ?F ?X “ versus “
          ?G ?X “ in worlds “ ?N “I” ?M
;;   МЕЖМИРОВОЕ ПРОТИВОРЕЧИЕ: “ ?F ?X “ противоречит “
;;   ?G ?X “ в мирах “ ?N “I” ?M
          t crlf)
     (modify ?W (task contra) (prior ?M))
)
```

Обратим внимание на то, что вместе с фиксацией самого факта противоречия в последнем правиле фиксируется и идентификатор ранее покинутого «мира», с которым конфликтует текущий. Эта информация потребует для выполнения отката с помощью правил, которые будут представлены в следующем разделе.

Подготовка рабочей памяти к выполнению отката

При подготовке к откату нужно выполнить одну из двух возможных операций с рабочей памятью:

- если к противоречию привел выбор определенного дизъюнкта, нужно удалить контекст, созданный в результате этого выбора;
- если к противоречию привело предположение о правдивости персонажа, нужно удалить соответствующий контекст и проанализировать, к чему приведет противоположное предположение.

В любом случае из рабочей памяти нужно удалить объекты claim. Поскольку откат никогда не выполняется в противоречивом контексте, такое удаление не повлияет на полноту представления информации о задаче. Если потребуется вновь сформировать удаленный контекст при других предположениях, это можно будет выполнить, повторив вычисления с «чистого листа».

```
;; ЕСЛИ обнаружено противоречие с одним из
;; дизъюнктивных контекстов «мира»,
;; ТО удалить все утверждения (объекты claim)
;; этого контекста.
;; ПРИМЕЧАНИЕ: правило будет активизироваться повторно,
;; пока не будут удалены все ненужный объекты.
(defrule clean-context
     (declare (salience 50))
     (world (tag ?N) (task contra) (prior 0)
          (context ?S&~0))
     ?F <- (claim (reason ?N) (context ?S))
=>
     (retract ?F)  )
)
;; ЕСЛИ противоречие обнаружено в текущем “мире” в
;; предположении о правдивости,
;; ТО повторить анализ, предположив лживость персонажа.
(defrule switch-context
     (declare (salience 40))
;; Если больше нет правых дизъюнктов,
```

```
     ?W <- (world (tag ?N) (scope truth) (task contra)
          (prior 0) (context ?S&~1))
=>
;; изменить предположение и сформировать новый контекст.
     (modify ?W (scope falsity) (task check) (context 0))
)
;; Удалить все утверждения (объекты claim),
;; сформированные на основании предположения о
;; правдивости.
;; ПРИМЕЧАНИЕ: правило будет активизироваться повторно,
;; пока не будут удалены все ненужные объекты.
(defrule sweep-truth
     (declare (salience 100))
     (world (tag ?N) (scope falsity))
     ?F <- (claim (reason ?N) (scope truth))
=>
     (retract ?F)
)
```

Последнее правило демонстрирует, как с помощью полей reason и scope можно отслеживать объекты claim. В данной программе используется тот же прием, что и в системах обработки правдоподобия.

Теперь можно приступить к разработке правил, выполняющих откат.

Выполнение отката

Каждое отдельное высказывание в задачах рассматриваемого класса формирует свой «мир». Поскольку высказывания обрабатываются последовательно, возможна ситуация, когда «мир», который программа анализирует в текущий момент, вступает в конфликт с одним из «миров», обработанных ранее. Дальнейшие действия в такой ситуации зависят от того, на каких предположениях основаны конфликтующие «миры».

Если текущий «мир» был проанализирован только в предположении о правдивости соответствующего персонажа, то, очевидно, нужно повторить его анализ, но уже в предположении, что персонаж – лжец. Если и в этом случае конфликтная ситуация сохраняется, следовательно, мы исчерпали все возможности ее локального разрешения, поэтому нужно подумать о возврату к анализу ранее покинутого «мира» и повторить его анализ, но уже на основе других предположений.

Но возврат имеет смысл выполнять только в том случае, если в ранее покинутом «мире» не были проанализированы все возможные сочетания предположений и контекстов. Таким образом, одно из условий, при которых целесообразно выполнять откат, состоит в том, что покинутый «мир» был проанализирован только в предположении правдивости соответствующего персонажа и что есть еще

возможность проанализировать следствия из противоположного предположения.

Правило undirected-falsity выполняет необходимые для этого подготовительные операции. Смысл слова undirected (ненаправленный) состоит в том, что это правило реализует откат в «хронологическом» порядке создания «миров». В механизме разрешения конфликтов в CLIPS реализовано «хронологическое предпочтение», которое обеспечивает откат к последнему из ранее сформированных «миров», удовлетворяющих заданным условиям. Но при этом не предпринимается никакой попытки локализовать в процессе выбор точки, которая стала причиной конфликта, например выбрать именно тот «мир», с которым конфликтует текущий.

;

```
; Хронологический откат к тому “миру”, который был
;; покинут без выполнения анализа в предположении
;; о лживости (поле scope содержит значение truth,
;; а поле task – значение check).
(defrule undirected-falsity
     (world (tag ?N) (scope falsity) (task contra))
     ?W <- (world (tag ?M&: (< ?M ?N))
          (scope truth) (task check))
=>
     (modify ?W (task back))
)
```

Альтернативный сценарий используется в ситуации, когда в ранее покинутом «мире» не были проанализированы все дизъюнкты составного утверждения. «Мир» был покинут, когда обнаружилось, что проанализированный дизъюнкт не противоречит предположению, поэтому прочие дизъюнкты просто не рассматривались. Теперь, когда обнаружилось противоречие с другим «миром», можно вновь вернуться к ранее незавершенному анализу и попробовать, не разрешится ли конфликт в результате исследования другого дизъюнкта.

Приведенное ниже правило undirected-disjunct выполняет подготовку к такому откату в хронологическом порядке.

```
;; Хронологический откат к тому «миру», который был
;; покинут без завершения анализа дизъюнктов.
(defrule undirected-disjunct
     (world (tag ?N) (scope falsity) (task contra))
     V <- (world (tag ?M&: (< ?M ?N)) (task check)
          (context 1))
     claim (content OR ?P ?X ?Q ?Y) (reason ?M)
          (scope ?S))
=>
;; Дизъюнкт в ранее покинутом “мире”, анализ которого
;; не был выполнен.
     assert (claim (content ?Q ?Y) (reason ?M) (scope ?S)
          (context 2)))
;; Зафиксировать необходимость отката в этот “мир”.
     Modify ?V (task back))
)
```

Хронологический откат является не единственной операцией такого рода. Ниже представлены «направленные» (directed) версии соответствующих правил, в которых используется информация о том, в каком именно «мире» имеется утверждение, ставшее причиной конфликта с текущим «миром». Эта информация содержится в слоте prior текущего объекта world.

```
;; Если обнаружено противоречие между
;; объектами world M и N
;; и объект M создан ранее объекта N,
;; причем анализ M в предположении о лживости
;; соответствующего высказывания не был выполнен,
;; ТО вернуться к анализу объекта M.
(defrule directed-falsity
     (world (tag ?N) (scope falsity) (task contra)
          (prior ?M&”0))
     ?W <- (world (tag ?M) (scope truth) (task check))
=>
     modify ?W (task back))
)

;; Если обнаружено противоречие между
;; объектами M и N
;; и объект M создан ранее объекта N, причем
;; не был выполнен анализ всех дизъюнктов в M,
;; ТО вернуться к анализу объекта M.
(defrule directed-disjunct
```

```
      (world (tag ?N) (scope falsity) (task contra)
           (prior ?M&~0))
      ?V <- (world (tag ?M) (task check) (context 1))
      (claim (content OR ?P ?X ?Q ?Y) (reason ?M)
           (scope ?S))
=>
;; Дизъюнкт в ранее покинутом “мире”, анализ которого
;; не был выполнен.
      (assert (claim (content ?Q ?Y) (reason ?M)
           (scope ?S) (context 2)))
;; Зафиксировать необходимость отката в этот “мир”.
      (modify ?V (task back))
)
```

Если вы думаете, что эти два правила позволяют справиться со всеми возможными ситуациями, в которых может возникнуть необходимость выполнить откат, то вы ошибаетесь. «Миры» W и V могут конфликтовать, хотя в обоих проанализированы все варианты предположений и все дизъюнкты. А источник конфликта при этом находится в некотором третьем «мире», в котором не был завершен анализ предположений или дизъюнктов.