Курс лекции по дисциплине YaPZ7302 Языки представления знаний

Лекция 3. Правила

Цель: Изучение синтаксиса правил.

План:

1. Понятие «правила»
2. Создание правил. Конструктор *defrule*
3. Пример. Программа "Hello-World!"
4. Основной цикл выполнения правил
5. Свойства правил

CLIPS поддерживает эвристическую и процедурную парадигму представления знаний. Для представления знаний в процедурной парадигме CLIPS предоставляет такие механизмы, как глобальные переменные, функции и родовые функции. Мы рассмотрим такой способ представления знаний, как *правила.* Правила в CLIPS служат для представления эвристик или так называемых "эмпирических правил", которые определяют набор действий, выполняемых при возникновении некоторой ситуации. Разработчик экспертной системы определяет набор правил, которые вместе работают над решением некоторой задачи. Правила состоят из *предпосылок* и *бедствия.* Предпосылки называются также *ЕСЛИ-частъю правила, левой частью правила* или *LHS правила* (left-hand side of rule). Следствие называется *ТО-частью правила, правой частью правила* или *RHS правила* (right-hand side of rule).

Предпосылки правила представляют собой набор условий (или условных элементов), которые должны удовлетвориться, для того чтобы правило выполнилось. Предпосылки правил удовлетворяются в зависимости от наличия или отсутствия некоторых заданных фактов в списке фактов (о котором было рассказано в предыдущей главе) или некоторых созданных объектов, являющихся экземплярами классов, определенных пользователем. Один из наиболее распространенных типов условных выражений в CLIPS — *образцы* (patterns). Образцы состоят из набора ограничений, которые используются для

определения того, удовлетворяет ли некоторый факт или объект условному элементу. Другими словами, образец задает некоторую маску для фактов или объектов. Процесс сопоставления образцов фактам или объектам называется *процессом сопоставления образцов* (pattern-matching). CLIPS предоставляет механизм, называемый *механизмом логического вывода* (inference engine), который автоматически сопоставляет образцы с текущим списком фактов и определенными объектами в поисках правил, которые применимы в данный момент.

Следствие правила представляется набором некоторых действий, которые необходимо выполнить, в случае если правило применимо к текущей ситуации. Таким образом, действия, заданные вследствие правила, выполняются по команде механизма логического вывода, если все предпосылки правила удовлетворены. В случае если в данный момент применимо более одного правила, механизм логического вывода использует так называемую *стратегию разрешения конфликтов* (conflict resolution strategy), которая определяет, какое именно правило будет выполнено. После этого CLIPS выполняет действия, описанные вследствие выбранного правила (которые могут оказать влияние на список применимых правил), и приступает к выбору следующего правила. Этот процесс продолжается до тех пор, пока список применимых правил не опустеет.

Чтобы лучше понять сущность правил в CLIPS, их можно представить в виде оператора IF-THEN, используемого в процедурных языках программирования, например, таких как Ada или C. Однако условия выражения IF-THEN в процедурных языках вычисляются только тогда, когда поток управления программы непосредственно попадает на данное выражение путем последовательного перебора выражений и операторов, составляющих программу. В CLIPS, в отличие от этого, механизм логического вывода создает и постоянно модифицирует список правил, условия которых в данный момент удовлетворены. Эти правила запускаются на выполнение механизмом логического вывода. С этой стороны правила похожи на обработчики сообщений, присутствующие в таких языках программирования, как, например, Ada или Smalltalk.

Без правил не обойдется ни одна экспертная система, так что правила и язык их представления в экспертной системе можно смело назвать самой важной частью любой экспертной оболочки. Успех экспертной системы во многом определяется тем, насколько удачен способ представления знаний в виде правил, и насколько хорошо им владеет разработчик экспертной системы. Вся данная глава посвящена правилам, их синтаксису и способам построения, их функционированию и назначению, а также приемам их применения.

Создание правил. Конструктор *defrule*

Для добавления новых правил в базу знаний CLIPS предоставляет специальный конструктор defrule. В общем виде синтаксис данного конструктора можно представить следующим образом:

Определение. Синтаксис конструктора defrule

(defrule
<имя-правила>
[<комментарии>]
[<определение-свойства-правила>]
<предпосылки > ; левая часть правила
=>
<следствие> ; правая часть правила
)

Имя правила должно быть значением типа symbol. В качестве имени правила нельзя использовать зарезервированные слова CLIPS, которые были перечислены ранее. Повторное определение существующего правила приводит к удалению правила с тем же именем, даже если новое определение содержит ошибки.

Комментарии являются необязательными, и, как правило, описывают назначения правила. Комментарии необходимо заключать в кавычки. Эти комментарии сохраняются и в дальнейшем могут быть доступны при просматривании определения правила.

Определение правила может содержать объявление свойств правила, которое следует непосредственно после имени правила и комментариев. Более подробно свойства правила будут рассмотрены ниже.

В справочной системе и документации по CLIPS для обозначения предпосылок правила чаще всего используется термин "LHS of rule", а для обозначения следствия "RHS of rule", поэтому в дальнейшем мы будем использовать аналогичную терминологию — левая и правая часть правила.

Левая часть правила задается набором условных элементов, который обычно состоит из условий, примененных к некоторым образцам. Заданный набор образцов используется системой для сопоставления с имеющимися фактами и объектами. Все условия в левой части правила объединяются с помощью неявного логического оператора and. Правая часть правила содержит список действий, выполняемых при активизации правила механизмом логического вывода. Для разделения правой и левой части правил используется символ =>. Правило не имеет ограничений на количество условных элементов или действий. Единственным ограничением является свободная память вашего компьютера. Действия правила выполняются последовательно, но тогда и только тогда, когда все условные элементы в левой части этого правила удовлетворены.

Если в левой части правила не указан ни один условный элемент, CLIPS автоматически подставляет условие-образец initial-fact или initial-object. Таким образом, правило активизируется всякий раз при появлении в

базе знаний факта initial-fact ИЛИ Объекта initial-object. О факте initial-fact было рассказано в предыдущей главе, об объекте initial-object вы узнаете в *гл. 11.* Если в правой части правила не определено ни одно действие, правило может быть активировано и выполнено, но при этом ничего не произойдет.

Более подробно синтаксис левой и правой части правил будет описан далее в этой главе, а пока, в качестве демонстрации применения правил, напишем простейшую CLIPS-программу, которая по традиции здоровается со всем

миром, сразу после своего рождения. Вы наверно знакомы с текстом подобных программ для процедурных языков программирования, таких как, например, C. На языке CLIPS такая программа будет выглядеть следующим образом:

Пример. Программа "Hello-World!"

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

Так как это первая наша программа на языке CLIPS, разберем подробно все ее действия и то, каким образом они выполняются.

Функция clear полностью очищает систему, т. е. удаляет все правила, факты и прочие объекты базы знаний CLIPS, добавленные конструкторами, приводит систему в начальное состояние, необходимое для каждой новой программы.

Затем, с помощью конструктора defrule в систему добавляется новое правило с именем Hello-World и соответствующими комментариями. Левая часть правила в данном случае отсутствует, поэтому CLIPS автоматически формирует предпосылки, состоящие из единственного условного выражения (initial-fact). Это выражение является образцом простейшего типа. При запуске программы на выполнение механизм логического вывода CLIPS будет искать в списке фактов факт (initial-fact) и если он там будет найден — активизирует правило.

Правая часть нашего правила состоит из нескольких вызовов функции printout. Сейчас же вам необходимо знать только то, что эта функция выводит текстовое выражение в один из потоков вывода. Параметр t задает стандартный поток вывода — экран. Он аналогичен, например, стандартному потоку cout в C++. Выражение clrf служит для перехода на новую строку.

Функция reset, как уже упоминалось ранее, очищает список фактов и заносит в него факт (initial-fact), что очень важно для нормального функционирования нашей программы. И, наконец, функция run запускает механизм логического вывода и приводит нашу программу в движение. Если описанные выше действия были выполнены правильно, то вы должны увидеть результат, аналогичный приведенному на рис. 27 — фразу "HELLO WORLD!!!" в красивой рамочке из звездочек:

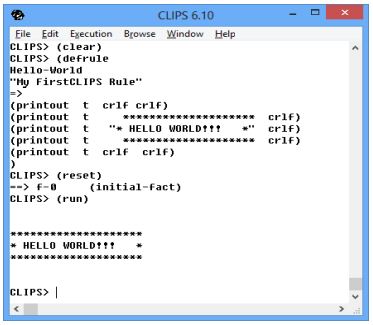

Рис. 27. Результат работы программы "Hello-World!"

Чтобы запустить нашу программу на выполнение еще раз, достаточно вызвать функции reset и run. Эти функции можно вводить с клавиатуры, кроме того, они доступны в меню **Execution** и имеют "горячие" клавиши <Ctrl>+<E> и <Ctrl>+<R> соответственно.

CLIPS поддерживает ряд функций, команд и визуальных средств, необходимых для эффективной работы с правилами. Самые основные из них будут рассмотрены в данной главе по мере необходимости. Сейчас же рассмотрим визуальный инструмент, доступный пользователям Windows-версии среды CLIPS — **Defrule Manager** (Менеджер правил). Для запуска менеджера правил в меню **Browse** выберите пункт **Defrule Manager.** Внешний вид этого инструмента показан на рис. 28.

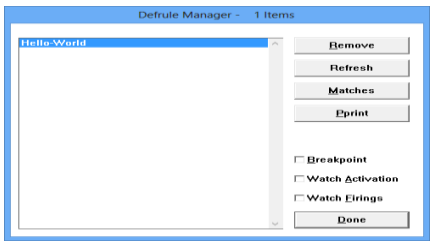

Рис. 28. Окно менеджера правил

Менеджер отображает список правил, присутствующих в системе в данный момент, и позволяет выполнять над ними ряд операций. Например, с помощью кнопки **Remove** можно удалить выбранное правило из системы, а с помощью **Pprint** вывести в окне CLIPS определение выделенного правила вместе с введенными комментариями. Общее количество правил отображается в заголовке окна менеджера — **Defrule Manager — 1 Items.** Другие возможности менеджера правил будут рассмотрены позже, по мере необходимости.

Как вы уже могли убедиться, разбирая приведенный выше пример несложной программы, довольно тяжело создать даже минимально полезную программу на языке CLIPS, не представляя себе, что именно происходит при выполнении вашей программы. При создании экспертных систем необходимо точно знать, как происходит сопоставление образцов, заданных в левой части правила, каким именно образом выбирается правило для выполнения и т. д. Именно поэтому в последующих разделах мы остановимся на внутренних алгоритмах представления и обработки правил в CLIPS, а уже затем перейдем к описанию функций и команд, связанных с правилами.

Основной цикл выполнения правил

После того как в систему добавлены все необходимые правила и приготовлены начальные списки фактов и объектов, CLIPS готов выполнять правила. В традиционных языках программирования точка входа, точка остановки и последовательность вычислений явно определяются программистом. В CLIPS поток исполнения программы совершенно не требует явного определения. Знания (правила) и данные (факты и объекты) разделены, и механизм логического вывода, предоставляемый CLIPS, применяет данные к знаниям, формируя *список применимых правил,* после чего последовательно выполняет их.

Этот процесс называется *основным циклом выполнения правил* (basic cycle of rule execution). Рассмотрим последовательность действий (шагов), выполняемых системой CLIPS в этом цикле в момент выполнения нашей программы:

1. Если был достигнут предел выполнения правил или не был установлен текущий фокус, выполнение прерывается. В противном случае, для выполнения выбирается первое правила модуля, на котором был установлен фокус. Если в текущем плане выполнения нет удовлетворенных правил, то фокус перемещается по стеку фокусов и устанавливается на следующий модуль в списке. Если стек фокусов пуст, выполнение прекращается. Иначе шаг 1 выполняется еще один раз. Более подробно о модулях и фокусе будет рассказано в *гл. 12.*

2. Выполняются действия, описанные в правой части выбранного правила. Использование функции return может менять положение фокуса в стеке фокусов. Число запусков данного правила увеличивается на единицу, для определения предела выполнения правила.

3. В результате выполнения шага 2 некоторые правила могут быть активированы или дезактивированы. Активированные правила (т. е. правила, условия которых удовлетворяются в данный момент) помещаются в план решения задачи модуля, в котором они определены. Размещение в плане определяется приоритетом правила (salience) и текущей стратегией раз решения конфликтов (эти понятия будут описаны ниже). Дезактивированные правила удаляются из текущего плана решения задачи. Если для правила установлен режим просмотра активаций, то пользователь получит соответствующее информационное сообщение при каждой активации или дезактивации правила (режим просмотра активаций можно установить с помощью диалогового окна **Watch Options.** Для этого выберите пункт **Watch** в меню **Execution** и установите флажок **Activations).**

Если установлен режим динамического приоритета (dynamic salience), то для всех правил из текущего плана решения задачи вычисляются новые значения приоритета. После этого цикл повторяется с шага 1.

Свойства правил

Для более полного понимания материала, изложенного далее в этой главе, необходимо разобраться с таким понятием, как свойства правил. Для задания свойства правила используется ключевое слово declare. Одно правило может иметь только одно определение свойства, заданное с помощью declare.

Определение . Синтаксис свойств правил

<определение-свойства-правила> ::= (declare <свойство-правила>)

<свойство-правила> ::= (salience <целочисленное выражение>) | (auto-focus TRUE | FALSE)

Свойство *salience*

Свойство правила salience позволяет пользователю назначать приоритет для своих правил. Объявляемый приоритет должен быть выражением, имеющим целочисленное значение из диапазона от —10 000 до +10 000. Выражение, представляющее приоритет правила, может использовать глобальные переменные и функции (которые будут описаны в *гл. 7* и *8* соответственно). Однако старайтесь не указывать в этом выражении функций, имеющих побочное действие. В случае если приоритет правила явно не задан, ему присваивается значение по умолчанию — 0.

Значение приоритета может быть вычислено в одном из трех случаев: при добавлении нового правила, при активации правила и на каждом шаге основного цикла выполнения правил. Два последних варианта называются *динамическим приоритетом* (dynamic salience). По умолчанию значение приоритета вычисляется только во время добавления правила. Для изменения этой установки можно использовать команду set-salience-evaluation.

Кроме того, пользователи Windows-версии среды CLIPS могут изменить эту настройку с помощью диалогового окна **Execution Options.** Для этого выберите пункт **Options** в меню **Execution,** в появившемся диалоговом окне укажите необходимый режим вычисления приоритета с помощью раскрывающегося списка **Salience Evaluation,** как показано на рис. 29.

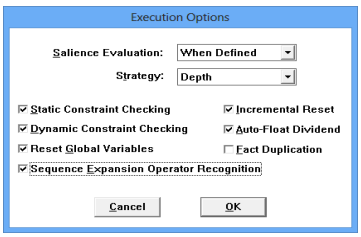

Рис. 29. Установка способа вычисления приоритетов правил

Замечание

Каждый метод вычисления приоритета содержит в себе предыдущий (т. е. если приоритет вычисляется на каждом шаге основного цикла выполнения правил, то он также вычисляется и при активации правила, а так же при его добавлении в систему).

Свойство *auto-focus*

Свойство auto-focus позволяет автоматически выполняться команде focus при каждой активации правила. Если свойство auto-focus установлено в значение TRUE, то команда focus в модуле, в котором определено данное правило, автоматически выполняется всякий раз при активации правила. Если свойству auto-focus присвоено значение FALSE, то при активации правила не происходит никаких действий. По умолчанию это свойство установлено в FALSE.

Контрольные вопросы:

1. Синтаксис конструктора defrule
2. Последовательность действий , выполняемых системой CLIPS.
3. Применение свойства salience.

Литература:

1. А. П. Частиков Т. А. Гаврилова Д. Л.Белов. РАЗРАБОТКА ЭКСПЕРТНЫХ СИСТЕМ. СРЕДА CLIPS. СПб: БХВ-Петербург, 2003

Ключевые слова:

Правила, *defrule*, цикл выполнения правил, свойства правил.