Лабораторная работа №2

по курсу «Интеллектуальные информационные системы»

Разработка прототипа ИС

Цель работы

Освоение основ проектирования ИС. Реализация структуры прототипа ИС в среде CLIPS.

Методические указания

Интеллектуальные системы

Современные интеллектуальные системы (ИС) – это системы, способные решать задачи в некоторой предметной области, традиционно считающиеся творческими. Как правило, они являются системами, основанными на знаниях (СОЗ), т.е. способны хранить знания, осуществлять рассуждение на основе правил вывода и пояснять получаемые результаты, а также производить самообучение. Прикладные ИС, моделирующие действия специалистов-экспертов при разрешении проблемных ситуаций в некоторой предметной области, называют ещё экспертными системами (ЭС). Общую схему ИС можно представить так, как показано на Рис. 1.

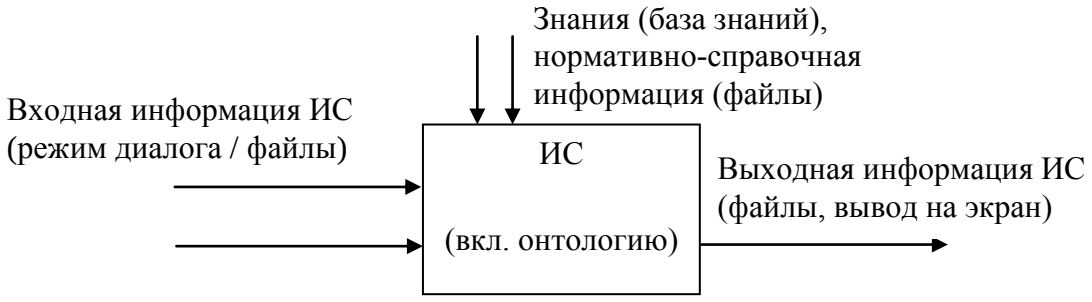

Рис. 1. Общая схема работы ИС.

Проектирование ИС

Процесс создания ИС имеет общие черты с процессом разработки ПО в целом, однако не полностью совпадает с ним. Для процесса разработки промышленной ИС выделяются следующие этапы:

1) выбор проблемы;
2) разработка прототипа ИС;
3) доработка до промышленной ИС;
4) оценка ИС;
5) стыковка ИС;

6) поддержка ИС.

Основная цель создания прототипа ИС заключается в оценке применимости выбранных методов инженерии знаний для решения обозначенных задач предметной области. Объем прототипа обычно составляет несколько десятков фреймов, правил или примеров, однако с демонстрационным прототипом уже могут работать не только члены коллектива разработчиков, но и пользователь, поскольку на этой стадии система способна решать часть реальных задач. В некоторых случаях в ходе дальнейшего создания ИС прототип может отбрасываться, но чаще демонстрационный прототип развивается в промышленную систему, прежде всего посредством расширения БЗ и разработки интерфейса пользователя.

В данной лабораторной работе вам предстоит реализовать прототип ИС с использованием специализированного языка CLIPS.

Краткое описание CLIPS

Язык CLIPS, который разрабатывался с 1985 г. космическим агентством NASA (США), в настоящее время находится в свободном доступе, а основное его назначение – создание интеллектуальных систем, в первую очередь на базе продукционной модели. Хотя в последнее время активного развития языка CLIPS как такового не происходит, на его основе были созданы такие средства как JESS (реализация языка разработки экспертных систем на Java) и FuzzyCLIPS (среда для разработки систем, использующих нечёткую логику), что подтверждает популярность среды CLIPS и адекватность основных подходов, заложенных в ней.

CLIPS сочетает в себе 3 парадигмы программирования: логическую, процедурную и объектно-ориентированную. Также, в CLIPS предусмотрены 3 основных формата представления информации: факты, глобальные переменные и объекты.

Среда CLIPSWindows реализует работу механизма вывода с правилами продукционной модели и обеспечивает (пусть и весьма ограниченный по возможностям) интерфейс пользователя ИС.

Работа с CLIPS

Интерфейс среды CLIPSWindows в целом соответствует Prolog. Команда **(run)** запускает выполнение программы в среде (с машиной логического вывода), **(reset)** – обновляет рабочую память, а **(clear)** – полностью очищает среду CLIPS (рабочую память). Опция главного меню **Load** позволяет загрузить файл с конструкциями, а **Load Batch** – файл с командами (программу).

Главный файл программы на CLIPS (например, *main.bat*) может иметь следующий вид:

```
(clear)

(load* "classes.clp") ; загрузка структуры классов
(load-instances "instances.clp") ; загрузка экземпляров классов
(load* "templates.clp") ; загрузка предопределённых фактов
(load* "functions.clp") ; загрузка функций
(load* "rules.clp") ; загрузка правил

(reset)
(run)
```

Вывод информации может осуществляться как на экран, так и в выходные файлы.

Например, нижеследующая функция может использоваться для запроса у пользователя значения, присвоения его переменной, а затем вывода его на экран:

```
(deffunction foo
        () ; функция не имеет аргументов
        (printout t crlf "Please input value:" crlf) ; t – ключ для вывода на экран
        (bind ?var (read)) ; bind – функция для присваивания значения
        (printout t crlf "Your value: " ?var crlf) ; crlf – символ конца строки
)
```

В свою очередь, команды *(save-facts ?file+name)* и *(save-instances ?file+name)* служат для сохранения всех имеющихся в CLIPS на текущий момент фактов и экземпляров классов соответственно. Значением переменной *?file+name* может быть, например, "system_output.clp".

Обратите внимание, что в CLIPS в именах функций для обозначения пробела обычно используется знак «-», а в именах переменных и строковых значениях – знак «+». Имя переменной всегда начинается со знака «?».

Форматы представления данных в CLIPS

Факты являются одной из основных форм высокого уровня для представления информации в системе CLIPS. Факт (**fact**) – это список элементарных значений, на которые ссылаются либо позиционно (упорядоченные (**ordered**) факты), либо по имени (неупорядоченные (**non-ordered**) или шаблонные (**template**) факты).

Простейшим фактом является факт из одного поля (single-field fact), который приблизительно соответствует булевской переменной в процедурных языках. Например:

(Im-on-duty-today) ; Обратите внимание, что в CLIPS в именах функций или фактах для обозначения пробела обычно используется знак «-», а в именах переменных и строковых значениях – знак «+».

Более сложными фактами являются факты с несколькими полями (multifield facts), которые приблизительно соответствуют переменной или массиву в процедурных языках:

(On-duty-today English Ivanov); Сегодня дежурный на уроке английского языка – Иванов. Обратите внимание, что порядок значений в записи факта – важен.

(On-duty-today English nil) ; В таком факте однажды определённая позиция не может быть просто пропущена, вместо неё должно стоять пустое значение nil (если дежурного сегодня нет).

(English Ivanov 4.5) ; Средний балл Иванова по английскому языку составляет 4.5. Хотя хорошим тоном является всё-таки начинать такой факт с названия отношения, как в следующем примере:

(Average-score English Ivanov 4.5)

Наконец, наиболее сложным типом фактов, соответствующем типу «запись» в Pascal и других языках, являются шаблонные (template) факты. Структура такого факта должна быть предварительно описана:

(deftemplate English-lesson ; Структура для фактов об уроке английского языка

(slot date (type STRING))

(slot day (allowed-symbols Monday Tuesday Wednesday Thursday Friday))

(slot on-duty-today) ; Кто сегодня дежурный (один студент)

(multislot students-present) ; Кто присутствует (мультислот – подразумевает несколько значений, т.е. более одного студента).

(slot teacher (default Salkova)) ; Преподаватель – по умолчанию, зав. кафедрой

)

Затем факт такого типа может быть задан, например, следующим образом:

(assert (English-lesson (date "20150209") (teacher Ryzhova) (day Monday) (on-duty Ivanov) (students-present nil))) ; Обратите внимание, что порядок задания слотов не важен, а некоторые из них могут иметь пустые значения

Каждый факт представляет часть информации и помещается в текущий список фактов (**fact-list**). Факты могут быть добавлены в список фактов (используя команду

assert), удалены из него (используя команду **retract**), изменены (используя команду **modify**) или скопированы (используя команду **duplicate**) в результате явного воздействия пользователя или при исполнении программы CLIPS. Для загрузки фактов из файла (добавления их в список фактов) используется команда **(load-facts)**, а для записи в файл всех существующих на данный момент в решателе фактов – команда **(save-facts)**. При этом рекомендуется указывать полный путь к файлам, например:

(save-facts "D:/work/facts.clp")

Глобальные переменные. Конструкция **defglobal** позволяет описывать переменные, которые являются глобальными в контексте окружения CLIPS. То есть глобальная переменная доступна в любом месте окружения CLIPS и сохраняет свое значение независимо от других конструкций. Напротив, некоторые конструкции (как, например, **defrule** или **deffunction**) могут иметь собственные локальные переменные, которые задаются в пределах описания конструкции. Обращение к таким переменным возможно только изнутри конструкции, где они описаны; за ее пределами они не имеют значения.

Функция **bind** используется, чтобы задать значения глобальным переменным. Значения глобальных переменных сбрасываются к начальным установочным значениям при выполнении команды окружения **reset** или если для глобальной переменной вызвана функция **bind** без соответствующего значения.

Классы. Познакомиться с объектно-ориентированной парадигмой, реализованной в CLIPS, вам предстоит в лабораторной работе №3.

Правила в CLIPS

Среда CLIPS имеет встроенные средства для осуществления логического вывода, но позволяет программисту управлять очередностью выполнения правил (посредством параметра *salience*).

Эвристический подход. Одним из основных подходов к представлению знаний в CLIPS является использование **правил**. Разработчик интеллектуальной системы (инженер по знаниям) задает набор правил, которые совместно работают над разрешением проблемы. Правило состоит из антецедента (это есть ни что иное как часть "ЕСЛИ...", т.е. список условий, и называется **LHS**) и консеквента (часть "ТО...", т.е. список действий – **RHS**).

Правила в CLIPS реализованы в привычной человеку форме:

ЕСЛИ условие_1 и ... и условие_M удовлетворяются,
ТО
ВЫПОЛНИТЬ действие_1 и ... и действие_N.

Следует заметить, что количество условных предпосылок M и число действий N, подлежащих выполнению в случае удовлетворения условий, в общем случае не равны. Если КАЖДОЕ условие в LHS находит себя среди фактов, то происходит активизация правила и выполнение ВСЕХ действий, записанных в его RHS. В противном случае правило не активизируется. Для создания правила используется конструкция **defrule**.

```
(defrule start
        (initial-fact) ; начальный факт, всегда существующий в CLIPS
        =>
        (assert (Toyota+Corolla owned)) ; установление факта
)
```

Ещё примеры правил, записанных на языке CLIPS, для экземпляров и фактов (см. примеры dilemma1.clp и dilemma3.clp соответственно):

```
(defrule move-alone-instances
  ?node <- (object (is-a status)
                   (search-depth ?num)
                   (farmer-location ?fs))
  (object (is-a opposite-of) (value ?fs) (opposite-value ?ns))
  =>
  (duplicate-instance ?node (search-depth (+ 1 ?num))
                      (parent ?node)
                      (farmer-location ?ns)
                      (last-move alone)))


 (defrule move-alone-facts
  ?node <- (status (search-depth ?num)
                   (farmer-location ?fs))
  (opposite-of ?fs ?ns)
  =>
  (duplicate ?node (search-depth =(+ 1 ?num))
                   (parent ?node)
                   (farmer-location ?ns)
                      (last-move alone)))
```

Функции в CLIPS

CLIPS также поддерживает процедурный механизм, как большинство «традиционных» языков программирования, таких как Pascal или C. **Функции**, заданные конструкцией **deffunction** и встроенные функции CLIPS позволяют пользователю

создавать новые исполнимые элементы, выполняющие полезные второстепенные действия или возвращают некоторое полезное значение.

В CLIPS процедурный и эвристический механизмы представления знаний могут тесно взаимодействовать путем вызова пользовательских функций как из LHS, так и из RHS. для создания пользовательских функций используется конструктор **deffunction**.

Например, определим функцию **om(x,y)**, которая возвращает целую часть частного от деления переменной y на переменную x:

```
(deffunction om
    (?x ?y) ;; аргументы функции
    (div ?y ?x)
)
```

Обратите внимание, что в CLIPS **имя переменной** начинается с символа "?", что для вызова функции (в данном случае встроенной функции деления нацело **div**), используется префиксная нотация и что вся конструкция представляет собой список, состоящий из четырех полей. Необходимо также помнить, что конструкции языка CLIPS записываются в круглых скобках, и для вызова функции необходимо будет набрать, например, **(om 5 10)**.

Ход работы

1. Изучить методические указания к лабораторной работе.

2. Сформулировать цель создания ИС и согласовать её с преподавателем.

Предпочтительно, чтобы цель ИС (решаемая интеллектуальная задача) соответствовала предметной области, выбранной в рамках лабораторной работы №1:

1. Разработка ПО (ИС)
2. Анализ требований к ПО
3. Проектирование архитектуры ПО
4. Программная реализация ПО
5. Тестирование и отладка ПО
6. Внедрение и поддержка (сопровождение) ПО
7. Проектирование компьютерных интерфейсов
8. CASE-средства
9. Контроль качества ПО
10. Моделирование бизнес-процессов

По согласованию с преподавателем, студенты могут предложить свой вариант предметной области – например, соответствующий теме РГР.

3. Описать входную и выходную информацию для ИС, уровни значений входной информации.

Также необходимо разработать 2-3 примера работы системы (корректных сочетаний входной и выходной информации) и согласовать их с преподавателем.

4. Реализовать в среде CLIPSWindows прототип ИС, включающий в себя:

- Организацию ввода информации (опрос системой пользователя).
- Обработку информации при помощи правил и функций, для достижения цели создания ИС. Общее количество **правил и функций должно составлять не менее 10** (без учёта ввода-вывода данных).
- Структуру функций системы (2-3 реально работающих функции, для остальных – описание работы на естественном языке).
- Организацию вывода информации на экран и в файл.

Актуальная версия CLIPSWindows (на 2017 г. – 6.24 или 6.30) прилагается к лабораторной работе или доступна по адресу:
http://clipsrules.sourceforge.net/index.html

Реализуемая ИС должна основываться на знаниях, представленных в виде фактов или иных структур данных. Реализация жёсткой структуры правил в формате диалога (как в примерах, прилагаемых к лабораторной работе) **не является удовлетворительной**.

5. Оформить отчет по лабораторной работе.

Содержание отчёта:

- Краткое описание предметной области и цели ИС.
- Описание входной и выходной информации для ИС, 2-3 контрольных примера работы ИС.
- Описание правил, функций и классов.

Дополнительно к отчёту должны прилагаться файлы **работающей** программы (*.clp и др.).

6. Защитить лабораторную работу, при необходимости ответив на контрольные вопросы.

Контрольные вопросы

1. В чем заключается отличие переменных, фактов и экземпляров классов в CLIPS?

2. Какие команды используются для создания и удаления фактов в CLIPS?

3. Каким образом определяется очередность выполнения правил в CLIPS и для чего она может использоваться?

4. Что может находиться в LHS правил в CLIPS?

Литература

1. Официальные руководства пользователя и программиста по CLIPS (на английском языке):
http://clipsrules.sourceforge.net/OnlineDocs.html

2. Описание CLIPS (на русском языке):
http://masters.donntu.org/2005/kita/kapustina/ind/clips.htm

3. Примеры программ на CLIPS (от его автора, Gary Riley) – см. в папке *examples/*

4. Пример программы на CLIPS (на русском языке):
http://masters.donntu.org/2005/kita/kapustina/library/bz.htm

5. А.П. Частиков, Т.А. Гаврилова, Д.Л. Белов. Разработка экспертных систем. Среда CLIPS. Санкт-Петербург, «БХВ-Петербург», 2003 г.