ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Робототехника и мехатроника»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторной работы

«Разработка экспертной системы управления мобильным роботом в среде CLIPS 6.3»

Авторы
Тугенгольд А.К.,
Ивацевич Ю.Б.,
Юсупов А.Р.,
Иванов А.В.

Ростов-на-Дону, 2015

Аннотация

Методические указания к выполнению лабораторной работы «Разработка экспертной системы управления мобильным роботом в среде CLIPS 6.3» по дисциплинам «Искусственный интеллект в мехатронике» и «Искусственный интеллект в мехатронике и робототехнике» предназначены для студентов специальности 150306 «Мехатроника и робототехника» очной формы обучения.

Авторы

Док. техн. наук, профессор А.К. Тугенгольд

Канд. техн. наук, доцент Ю.Б. Ивацевич

Ст. преп. А.Р. Юсупов

Техник А.В. Иванов

Оглавление

Цель работы:

- получение практических навыков проектирования экспертных систем для решения задач планирования действий, системы управления мобильным роботом.

- ознакомление с основными принципами программирования на языке CLIPS;

- приобретение навыков работы в программной среде CLIPS 6.3;

- разработка программы по планированию действий технической системой в робототехнике и мехатронике.

Основные теоретические сведения

Порядок работы экспертной системы состоит из трех основных этапов:

1. сопоставление фактов и правил;
2. выбор правила, подлежащего активизации;
3. выполнение действий, предписанных правилом.

Основные элементы программирования в CLIPS

В системе CLIPS (C Language Integrated Production System) используется, диалект языка LISP, направленный на создание экспертных систем, используя синтаксис и стиль программирования схожий с человеческой речью и логикой мышления. CLIPS поддерживает объектно-ориентированную и процедурную парадигму программирования.

Язык CLIPS включает три основных элемента для написания программ:

- Простые типы данных;
- Конструкции для пополнения базы знаний;
- Функции для манипулирования данными.

Простые типы данных

В системе определено восемь базовых типов данных: Float (числа с плавающей точкой), integer (целочисленные), symbol (неразрывная последовательность символов), string (набор символов заключенный в двойные кавычки), external-address, fact-addres instance-name, instance-address.

Для записи чисел используются цифры (0-9), и символы (.), (+), (e).

Разделяющие символы: пробел, табуляция, двойные кавычки, (,), &, |, <, >, ;.
Символ (\) отменяет спец значение следом стоящего символа например:
("a and \"Ъ").

Работа с базой знаний. Факты

Факт представляет собой основную единицу данных (fact-list), используемую правилами. Каждому факту присваивается уникальный идентификатор факта. Индексы нумеруется с нуля.
Идентификатор факта имеет вид: f-n, где n – целое положительное число.

Пример: f-0 (today is Sunday) ;; факт№1 «сегодня воскресение»

f-1 (weather is warm) ;; факт№2 «погода теплая»

Факты представимы в двух форматах: позиционном и непозиционном.

В CLIPS существуют следующие зарезервированные слова, которые не могут использоваться как первое поле любого факта: test, and, or, not, declare, logical, object, exists, forall.

Непозиционные факты (шаблонные факты) - позволяют задавать имена каждому полю факта и затем обращаться к ним по имени.
Конструкция имеет общий вид:

(deftemplate <name>
(slot-1)
(slot-2)
…
(slot-N))

где <name> - имя шаблона, (slot-N) - именованное поле (или слот). Так же слоты могут быть ограничены по типу, значению, числовому диапазону, могут содержать значение по умолчанию.

Пример:

```
(deftemplate student ;;объявляем шаблон фактов с именем «student»
"a student record"      ;;строка с комментарием "запись о студенте"
(slot name (type STRING)) ;; слот содержит шаблон факта с
```

именем «name»
;;имеющий один аргумент строчного типа
(slot age (type NUMBER) (default 18))) ;;изначально, значение факта будет равно 18

Каждый слот начинается с ключевого слова slot или field. Слот включает поле данных, например name, тип данных, например STRING. Можно указать и значение по умолчанию (default).

Операции над фактами

В командной сроке CLIPS, чтобы добавить факт в fact-list (assert), удалить (retract), изменить (modify), дублировать (duplicate), вывести список фактов из памяти (facts), очистить базу фактов (clear), удалить массив фактов (undeffacts <list name>), подключить базу фактов (load), загрузить базу фактов в рабочую память (reset) нужно ввести соответствующую команду. Пример:

;;добавить факт «сегодня воскресение» в базу знаний
CLIPS> (assert (today is Sunday))
==>f-1 (today is Sunday) ;;факт успешно добавлен под индексом f-1
<Fact-1>

Команды, assert и retract, используются в программе для изменения базы фактов. Для использования базы фактов, нужно в начале ее подключить, затем загрузить в память. Команда reset сначала очищает базу фактов, а затем включает в нее факты из всех ранее загруженных массивов.

Работа с базой правил. Правила

В языке CLIPS правила имеют следующий формат:
(defrule <имя правила>
<предпосылка_1 >
< предпосылка_m >
=>
<действие_1 >
< действе_n >)

Параметр salience – хранит приоритет факта, принимает целочисленное значение в диапазоне (-10 000, 10 000). По умолчанию равен нулю.

Переменные объявляются конструкцией ?<name>. Если <name> - не заполнено, то вместо "?" может быть сопоставлен любой элемент.

Пример:

```
(defrule pick-a-chore ;;объявляем правило с именем «pick-a-chore»
"Choice days for the job" ;; комментарий «Выбор дней для работ»
;;если условия
(today is ?day)
(chore is ?job)
;;если условия буду сопоставлены с фактами
;;(today is Sunday) ;;сегодня – «воскресение»
;;(chore is carwash) ;;случайная работа – «автомойка»
=>
;; то в случае активизации оно включит в базу новый факт (do carwash on Sunday)
(assert (do ?job on ?day)) )
```

Аналогично, правило удаление факта из базы

```
(defrule drop-a-chore  ;;правило с именем «drop-a-chore»
(today is ?day) ;;если условия буду сопоставлено с фактом (today is Sunday)
;;в переменную ?chore сохраняется указатель на факт (do carwash on Sunday)
?chore <- (do ?job on ?day)
=>
(retract ?chore) ) ;;факт (do carwash on Sunday) удаляется из базы
```

где "<-" – оператор взятия адреса.

Функции для манипулирования данными. Определение функций

CLIPS поддерживает пользовательские и системные типы функций. Системные функции являются встроенными, а пользовательские определяются непосредственно в программе.

Математические и арифметические функции: +, -, *, /, ** (возведение в степень), Abs, Sqrt, Mod, Min, Max.

Пример 1:

(+ 2 5 8)

Логические операторы: | (ИЛИ), ~ (НЕ), & (И)

Оператор test служит для выполнения условий сравнения, с использованием логических операторов (or, not, and), в левой части правила. (test (or (> ?a 7) (< ?b -1))) ;; при ?a=8 и ?b=-2 – условие выполниться

Пользовательские функции объявляются командой deffunction.

(deffunction <имя функции> (<аргумент_1> ... < аргумент_n>)

<выражение1>

…

<выражение_n>)

Аргументами функции, могут быть данные простых типов, переменные или других функций.

Функция будет возвращать результат последнего выражения в списке.

Пример:

;; объявлена функция с именем between с аргументами ?lb, ?value, ?ub

(deffunction between (?l ?v ?r)

(or (> ?l ?v) (> ?v ?r)))

Эта функция возвращает true, если заданное целочисленное значение попало в диапазон между нижним и верхним пределами.

Чтобы присвоить переменной значение используется команда bind.

Пример: (bind ?a 4)

ПРАКТИЧЕСКАЯ ЧАСТЬ ЛАБОРАТОРНОЙ РАБОТЫ

Решение задач на планирование

Задача планировщика - определить последовательность действий модуля принятия решений, системы управления мобильным роботом.

Порядок выполнения

1. Изучить теоретический материал методических указаний.
2. Выполнить пример программы "Робот и ящик" в режиме отладки. (включить просмотр правил и фактов)
3. Модифицировать программу "Робот и ящик" таким образом, чтобы ящик необходимо было передвинуть в комнату С. Программу выполнять пошагово.
4. Составить программу на планирование действий робота согласно индивидуальному варианту задания.

Пример создания

Рассмотрим ситуацию когда есть две комнаты (А и В). В комнате «А» находится мобильный работ, а в комнате «В» расположен ящик. Задача робота, переместить ящик из «В» и «А».

Решение.

Зададим шаблон определяющий положение объекта:

(deftemplate in *;; имя шаблона «in»*
;; Object – имя объекта (ящик или робот)
;; location – название локации (комнаты)
(slot object (type SYMBOL)) *;;имеет символьный тип, принимаемого значения*
(slot location (type SYMBOL)))

Определим шаблон цели действия робота.

;; task – задача
;; action – имя действия, которое нужно выполнить
;; object – имя объекта, который нужно переместить
;; from – название локации (комнаты) из какой переместить
;; to – название локации (комнаты) в какую переместить

```
(deftemplate task
 (slot action (type SYMBOL))
 (slot object (type SYMBOL))
 (slot from (type SYMBOL))
```

(slot to (type SYMBOL)))

На основе шаблонов in и task запишем факты о начальном состоянии системы:

(deffacts world

(in (object robot) (location RoomA)) *;; робот находится в комнате A*

(in (object box) (location RoomB)) *;; ящик находится в комнате B*

;;задача: выполнить действие – перемещение ящика из комнаты A в B

(task (action transfer) (object box) (from RoomB) (to RoomA)))

Первые два факта задают начальное положение робота и ящика, третий определяет задачу. Далее зададим необходимые правила поведения робота для достижения цели. Весь процесс можно разбить на три этапа:

1. перемещение робота в комнату, где находится объект;
2. перемещение робота с объектом в комнату, определенную задачей;
3. завершение программы, если цель достигнута.

Опишем первое правило действия:

;; если ящик находится в другой комнате, то переместится туда.

(defrule move

(task (object ?X) (from ?Y))

(in (object ?X) (location ?Y))

;; если робот не в комнате Y, сохраняется ссылка на факт in,

?robot-position <- (in (object robot) (location ~?Y))

=>

;;*to modify* изменит значение поля *location*, факта *in*

(modify ?robot-position (location ?Y)))

;; т. е. робот перемещается в комнату, где находится объект, который необходимо переместить.

Теперь создадим правило перемещения робота с ящиком, в заданную комнату:

(defrule transfer

(task (object ?X) (from ?Y) (to ?Z)) *;;получение факта из базы*

?object-position <- (in (object ?X) (location ?Y)) *;;если ящик ?X находиться в комнате ?Y*

?robot-position <- (in (object robot) (location ?Y)) *;;если робот находиться в той же комнате*

=>

;; то переместить робота и объект в комнату ?Z

;;то есть изменить у обоих фактов значение поля location с ?Y на ?Z

(modify ?robot-position (location ?Z))

(modify ?object-position (location ?Z)))

Условием завершения работы будет являться наличие факта, что робот и ящик находятся к заданной комнате, указанной в слоте *to*.

(defrule stop

(declare (salience +1)) *;;повышаем приоритет правила, чтобы оно проверялось в первую очередь*

;; если значения полей факта task совпадают со значениями факта in

(task (object ?X) (to ?Y))

(in (object ?X) (location ?Y))

=>

(halt)) *;; halt – команда остановки программы*

Для выполнения программы необходимо сохранить код в файл с расширением (*.clp), загрузить программу для выполнения (load), очистить память и загрузить базу знаний (assert), запустить программу (run), по окончанию программы можно просмотреть итоговый список фактов (facts), либо в активировать просмотр фактов (Execution->Watch->Facts).

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

В помещение, разделённом на комнаты: A, B, C, D, E, находится робот и три ящика. Задача робота собрать все ящики и сложить из них башню в комнате, где изначально находиться робот. Исходные данные указаны в таблице 1.

Таблица 1

№ варианта	Начальное местоположение			
	робота	ящиков		
		1	2	3
1	A	A	B	C
2	B	D	E	A
3	C	B	C	D
4	D	E	A	B
5	E	C	D	E
6	A	A		C
7	B	D	E	
8	C		C	D
9	D	E		B
10	E	A	B	C
11	A	D	E	A
12	B	B		D
13	C	E	A	B
14	D		D	E
15	E	A	B	

**номер задания в списке соответствует порядковому номеру студента в списке группы. Варианты заданий распределяются циклично.*

Содержание отчета

В отчете необходимо указать: цель работы, описание индивидуального задания и код программы, результаты работы программы, выводы по работе.

Контрольные вопросы

1. Что такое система, основанная на знаниях?
2. Формы представления фактов?
3. Способ описания непозиционных фактов в CLIPS?
3. Какие механизмы представления знаний поддерживает CLIPS?
3. Что входит в понятие "Коэффициент уверенности"?
4. Что содержит параметр salience?
5. В чем главная особенность языков программирования экспертных систем от других языков программирования?

ЛИТЕРАТУРА, РЕКОМЕНДУЕМАЯ ПРИ ПОДГОТОВКЕ К ЛАБОРАТОРНОЙ РАБОТЕ

1. CLIPS a tool for building expert systems [электронный источник] - Режим доступа: http://clipsrules.sourceforge.net, свободный.

2. Джозеф Джарратано - Экспертные системы. Принципы разработки и программирование, 4е изд – 2007

3. В.Л. Афонин, В.А. Макушин – Интеллектуальные робототехнические системы. Курс лекций. Учебное пособие. Москва. 2005.

4. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. СПб.: Питер, 2001. с. 384.

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