动态模糊逻辑程序设计语言的指称语义

韩小芬 李凡长
(苏州大学计算机科学与技术学院 苏州 215006)

摘 要 文献[8]借鉴Dijkstra的监督命令程序结构,给出了动态模糊逻辑程序设计语言的基本框架结构。在此基础上,进一步扩充和完善,并根据指称语义的原理和方法,用结构归纳法给出动态模糊逻辑程序设计语言的指称语义,主要包括:动态模糊程序设计语言的语义域、语义函数及其指称语义。最后给出了一个动态模糊程序设计语言的例子以观察程序的运行过程。

关键词 动态模糊逻辑,动态模糊逻辑程序设计语言,指称语义

Denotational Semantics of Dynamic Fuzzy Logic Programming Language

HAN Xiao-fen LI Fan-zhang
(School of Computer Science & Technology, Soochow University, Suzhou 215006, China)

Abstract Reference[8] proposed the basic frame work of dynamic fuzzy logic programming language according to the guarded commands program structure proposed by Dijkstra. Based on it, this paper extended and completed the frame of dynamic fuzzy logic programming language, and then according to the principle and method of denotational semantics, defined its denotational semantics by structural induction, which includes the semantics domains and semantics functions of dynamic fuzzy logic programming language and its denotational semantics model. We provided an example to observe the execute process of the program.

Keywords Dynamic fuzzy logic, Dynamic fuzzy logic programming language, Denotational semantics

1 前言

语义学是语言设计中的重要内容之一,大致可以分为4个分支,即操作语义学、指称语义学、公理语义学和代数语义学[1]。

操作语义的基本思想是用抽象的方法描述语言中的每一成分的执行效果,以免所描述的语义依赖于该语言实现时所用的具体计算机。它是用一个抽象机定义一组抽象状态,把语言的语法表示成抽象的形式,然后指明抽象机每加工一个语言成分时对状态所做的改变[2,3]。指称语义学的研究首先是由D. Scott和C. Strachey等人开始的。它的基本思想是使语言的每一成分对应于一个数学对象,该对象即是语言成分的指称,它只是考虑各成分执行的最终效果,此效果是不依赖于其执行过程的[1,3]。公理语义学是在程序正确性验证的基础上发展起来的,它是采用逻辑方法对程序设计语言进行研究。它的基本思想是给出一种方法,使人们能在给定前提下验证某种特定的性质是否成立。因此公理语义方法是一个逻辑系统,它包括一组公理及其推理规则[1,2,4]。代数语义学的基本思想是把描述语义的逻辑系统和满足这个逻辑体系的模型区分开来。它是用代数方法来满足一个逻辑系统的各种模型,从而把一个模型的集合看成一个代数结构[2,5]。

文献[8]借鉴Dijkstra的监督命令程序结构,给出了动态模糊语言的框架及其操作语义。鉴于此方面的研究工作还不完善,本文进一步对该语言框架扩充和完善,通过指称语义描述方法,提出了动态模糊逻辑程序设计语言的指称语义模型,以期形成解决动态模糊性问题的程序设计方法。

在给出程序设计语言的形式语义时,我们着重于建立一个数学模型。目的是使其作为理解程序以及对程序的行为进行论证的基础。数学模型不仅对于各种各样的分析和验证是有用的,而且对刻画程序构造的含义显示出各种精妙细微之处,由此证明数学模型对语言设计是十分重要的[5,6]。

本文第2节给出了动态模糊逻辑的相关理论,同时给出了动态模糊程序设计语言的构成框架;第3节给出了动态模糊程序设计语言的语义函数,并定义了其指称语义;第4节给出了一个动态模糊程序设计的例子以让读者更好地理解动态模糊程序设计语言;最后对本文进行了总结并给出了下一阶段要做的工作。

2 动态模糊逻辑程序设计语言理论基础

2.1 动态模糊逻辑简介[9-11,16]

定义1 一个具有动态模糊性的语句称为动态模糊命题,简称DF命题,用大写字母A,B,C……表示。对于一个DF命题,一般没有绝对的真假,只能问它的DF真假度如何?

定义2 度量一个DF命题真假度用DF数$(\overleftarrow{a},\overrightarrow{a})\in[0,$

到稿日期:2008-04-30 本文的研究得到国家自然科学基金(60775045)和江苏省自然科学基金(BK2005027,BK2002040)资助。
韩小芬(1984—),女,硕士研究生,主要研究方向为动态模糊逻辑及其程序设计语言;李凡长(1964—),男,教授,博士生导师,CCF高级会员,主要研究方向为人工智能、动态模糊逻辑、机器学习、多Agent系统。

1]×[←,→]来表示，称为该命题的真假，常用小写字母($\overleftarrow{a}$,$\overrightarrow{a}$),($\overleftarrow{b}$,$\overrightarrow{b}$),($\overleftarrow{c}$,$\overrightarrow{c}$)……表示。

定义 3 一个 DF 命题可以看成在闭区间[0,1]×[←,→]取值的变量，称为 DF 命题变量，常用小写字母表示。

定义 4 设在论域 U 上定义一个映射：

($\overleftarrow{A}$,$\overrightarrow{A}$):($\overleftarrow{U}$,$\overrightarrow{U}$)→[0,1]×[←,→], ($\overleftarrow{u}$,$\overrightarrow{u}$)α($\overleftarrow{A}$($\overleftarrow{u}$),$\overrightarrow{A}$($\overrightarrow{u}$))记为($\overleftarrow{A}$,$\overrightarrow{A}$)=$\overleftarrow{A}$ 或$\overrightarrow{A}$，则称($\overleftarrow{A}$,$\overrightarrow{A}$)为($\overleftarrow{U}$,$\overrightarrow{U}$)上的动态模糊集，简称 DFS，称($\overleftarrow{A}$($\overleftarrow{u}$),$\overrightarrow{A}$($\overrightarrow{u}$))为隶属函数对($\overleftarrow{A}$,$\overrightarrow{A}$)的隶属度。

对于属于 D=[0,1]×[←,→]的元素，表明在某一时刻，其当前状态有两种发展趋势："→"指向好的或前进的趋势，"←"指向坏的或后退的趋势，并且这两种趋势的发展变化程度是模糊的而非精确的，用属于[0,1]之间的数进行描述。

2.2 动态模糊逻辑程序设计语言的基本数据类型的表示

数据类型，通俗地说是一个数据集合以及其上带有某些性质的操作集合。据此定义，DFL 程序设计语言的基本数据类型将是动态模糊集及其上的集合操作。动态模糊集的基本思想是把指称集映射到 D=[0,1]×[←,→]上。因此，DFL 程序设计语言的基本数据类型的表示可以看成是我们常见的高级程序语言的基本数据类型到[0,1]×[←,→]上的一个映射[12-15,17]。具体地说，DFL 程序设计语言的基本数据类型可表示如下：

(1) 动态模糊整型数据(DFInt)可以表示成(($\overleftarrow{i}$,$\overrightarrow{i}$),($\overleftarrow{d}$,$\overrightarrow{d}$))，其中 $i\in Integer, d\in D$。

(2) 动态模糊实型数据(DFReal)可以表示成(($\overleftarrow{r}$,$\overrightarrow{r}$),($\overleftarrow{d}$,$\overrightarrow{d}$))，其中 $r\in Real, d\in D$。

(3) 动态模糊字符型数据(DFChar)可以表示成(($\overleftarrow{c}$,$\overrightarrow{c}$),($\overleftarrow{d}$,$\overrightarrow{d}$))，其中 $c\in Char, d\in D$。

(4) 动态模糊布尔型数据(DFBool)可以表示成(($\overleftarrow{b}$,$\overrightarrow{b}$),($\overleftarrow{d}$,$\overrightarrow{d}$))，其中 $b\in Boolean, d\in D$。

其中，*Integer* 表示经典集合中的整型数据集，*Real* 表示经典集合中的实数型数据集，*Char* 表示经典集合中的字符型数据集，*Boolean* 表示经典集合中的布尔值集。

关于动态面模糊数据类型的运算操作并非本文工作的重点，若读者感兴趣具体可以参考文献[9-11]。

2.3 动态模糊逻辑程序设计语言的抽象语法

本文的 DFL 程序设计语言模型与文献[8]中是一致的，所以本文的程序设计语言的语句，原则上与常见的程序设计语言的语句一样，也主要有赋值语句、条件语句、循环语句和输入输出语句，但是在格式和内容的细节处理上又有所不同。因为本文的 DFL 程序设计语言与常见的程序设计语言的重要区别是它可以处理动态模糊数据。要做到这一点我们可以借鉴 Dijkstra 所提出的监督命令的程序结构，这种程序结构的特点是在程序中引入了不确定性，为每个可能被执行的语句提供一个监督条件。我们试图对这种程序结构进行变形以引入动态模糊性。监督命令程序结构的形式如下：

〈监督命令〉::=〈监督条件〉→〈监督语句表〉

〈监督条件〉::=〈布尔表达式〉

〈监督语句表〉::=〈监督语句〉{;〈监督语句〉}

〈监督语句〉::=〈条件语句〉|〈循环语句〉|〈语句〉

〈条件语句〉::=IF〈监督命令集〉FI

〈循环语句〉::=DO〈监督命令集〉OD

〈监督命令集〉::=〈监督命令〉{□〈监督命令〉}

其中符号"□"表示有限选择，即从"□"隔开的多个监督命令中任选一个执行。从上面的形式中我们可以看出监督命令的特殊之处在于对不确定性的处理，主要表现在它的条件语句和循环语句上。

这里的条件语句的一般形式是

IF $g_1\rightarrow s_1$ □ $g_2\rightarrow s_2$ □……□ $g_n\rightarrow s_n$ FI

执行时任选一个监督条件 g_i 为真的监督语句表执行之，执行完毕即离开此条件语句转入下一个语句。若所有的 g_i 均不成立，即认为是出现了错误情况。

循环语句的一般形式是

DO $g_1\rightarrow s_1$ □ $g_2\rightarrow s_2$ □……□ $g_n\rightarrow s_n$ OD

执行时任选一个监督条件 g_i 为真的监督语句表执行之，执行完毕后重复上述过程，直到所有 g_i 均不成立，才离开循环语句并进入下一个语句。

上面我们给出了监督命令程序结构的形式，下面我们仿照这种程序结构形式给出 DFL 程序设计语言的语法结构。我们知道动态模糊集是由指称集到[0,1]×[←,→]的映射来定义的，为了引入动态模糊集我们有必要增加一个特殊 D=[0,1]×[←,→]，程序中的每个变量通过映射对应到 D 中的一个元素，以描述动态模糊隶属度。下面我们通过 BNF 来描述 DFL 程序设计语言的语法。为了描述的方便，我们省去一些不必要的细节，这里只给出抽象语法而不是它的具体语法。我们关心的是程序设计语言的含义而不是如何描述它们的理论，所以对我们来说，抽象语法已经满足要求了。

DFL 语言的语法域有[1]：

p:Program	程序
e:Exp	表达式
s:State	语句
d:Declare	声明
t:Type	类型
ide:Identifiers	标识符
op:Exp_operators	操作
Bop:Bool_operators	操作
DFInt:DFConst	DF 整型常量
DFReal:DFConst	DF 实型常量
DFChar:DFConst	DF 字符型常量
DFBool:DFConst	DF 布尔型常量

我们用元变量来表示语法集合的元素。如 s 表示语句集 S 中的元素，e 表示算术表达式 E 中的元素。为了描述的方便，在 *DFL* 语言的语法中我们约定几个特定的字母[6]：

s 表示语句集 S 中的元素；

e 表示算术表达式集 E 中的元素；

g 表示监督命令集 G 中的元素；

b 表示布尔表达式集 B 中的元素。

动态模糊语言完整的构造规则如下[1,15,17]：

p::=d ;s

d::=t:Ide|d′;d″

t::= DFInt| DFReal| DFChar| DFBool

s::=skip|abort|(($\overleftarrow{x}$,$\overrightarrow{x}$),($\overleftarrow{d}$,$\overrightarrow{d}$)):=e|(s; s)|if g fi|do g od

e::= −e|(($\overleftarrow{x}$,$\overrightarrow{x}$),($\overleftarrow{d}$,$\overrightarrow{d}$))|e_0 op e_1|(e) | (($\overleftarrow{n}$,$\overrightarrow{n}$),($\overleftarrow{d}$,$\overrightarrow{d}$))

op::=+|-|*|/|%

g::=b→s|(g_0□g_1)

b::=(($\overleftarrow{true},\overrightarrow{true}$),($\overleftarrow{d},\overrightarrow{d}$))

|(($\overleftarrow{false},\overrightarrow{false}$),($\overleftarrow{d},\overrightarrow{d}$))

|b_0 Bop b_1|e_0 Rel e_1

Bop::=&|||!=

Rel::==|>|<|>=|<=|!=

其中 abort 表示从任一初始状态都不会产生终止状态。

($\overleftarrow{x},\overrightarrow{x}$),($\overleftarrow{d},\overrightarrow{d}$))$\in Var$ 是变量，其中

Var={(($\overleftarrow{x},\overrightarrow{x}$),($\overleftarrow{d},\overrightarrow{d}$))|($\overleftarrow{x},\overrightarrow{x}$)∈DFInt}∪{(($\overleftarrow{x},\overrightarrow{x}$),($\overleftarrow{d},\overrightarrow{d}$))|($\overleftarrow{x},\overrightarrow{x}$)∈DFReal}；

($\overleftarrow{n},\overrightarrow{n}$),($\overleftarrow{d},\overrightarrow{d}$))是数值常量，即

($\overleftarrow{n},\overrightarrow{n}$),($\overleftarrow{d},\overrightarrow{d}$))∈{($\overleftarrow{n},\overrightarrow{n}$),($\overleftarrow{d},\overrightarrow{d}$))|($\overleftarrow{n},\overrightarrow{n}$)∈DFInt}∪{(($\overleftarrow{n},\overrightarrow{n}$),($\overleftarrow{d},\overrightarrow{d}$))|($\overleftarrow{n},\overrightarrow{n}$)∈DFReal}($\overleftarrow{d},\overrightarrow{d}$)∈D 是动态模糊隶属度。

3 动态模糊程序设计语言指称模型

3.1 DFL 语言的语义域

第 2 部分所设计语言的语义域为：

动态模糊整型数据：

DFInt={(($\overleftarrow{i},\overrightarrow{i}$),($\overleftarrow{d},\overrightarrow{d}$))|$i\in Integer, d\in D$}动态模糊实型数据：

DFReal={(($\overleftarrow{r},\overrightarrow{r}$),($\overleftarrow{d},\overrightarrow{d}$))|$r\in Real, d\in D$}动态模糊字符型数据：

DFChar={(($\overleftarrow{c},\overrightarrow{c}$),($\overleftarrow{d},\overrightarrow{d}$))|$c\in Char, d\in D$}

动态模糊布尔值：

DFBool={(($\overleftarrow{b},\overrightarrow{b}$),($\overleftarrow{d},\overrightarrow{d}$))|$b\in Boolean, d\in D$}

基值：$B=I+$DFBool

运算：$F=F_a+F_b+F_r$

算术运算：$F_a=F_{au}+F_{ab}$

一元算术运算：F_{au}=DFInt→DFInt

二元算术运算：

+，−，*，/，%：F_{ab}=DFInt×DFInt→DFInt

布尔运算：$F_b=F_{bu}+F_{bb}$

一元布尔运算：

!：F_{bu}=DFBool→DFBool

二元布尔运算：

&，|：F_{bb}=DFBool×DFBool→DFBool

关系运算：

=，>，<，>=，<=，!=，：

F_r=DFInt×DFInt→DFBool

指称值：δ：D= DFInt+ DFBool+F

表示值：v：E=D

σ：$S=L\to V$

ρ：$U=Ide\to D$

3.2 DFL 语言的语义函数

根据上文的分析，我们用结构归纳法定义动态模糊逻辑程序设计语言的语义函数如下：

S：S exp→($\Sigma\to\Sigma$)

E：E exp→($\Sigma\to Var$)

G：G exp→($\Sigma\to\Sigma$)

B：B exp→($\Sigma\to DFBool$)

O：OPr→F

3.3 DFL 语言的指称语义

我们在各种语句前面加上其所属类别的符号代码，以加强直观性和可理解性，即 E 表示算术表达式，S 表示语句，G 表示监督命令，B 表示布尔表达式[6]。

1. 算术表达式的指称模型

首先，用结构归纳法把算术表达式的指称定义为状态和数之间的关系：

(1)$E[|(\overleftarrow{n},\overrightarrow{n}),(\overleftarrow{d},\overrightarrow{d}))|]=\{(\sigma,(\overleftarrow{n},\overrightarrow{n}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\}$

(2)$E[|(\overleftarrow{x},\overrightarrow{x})(\overleftarrow{d},\overrightarrow{d}))|]=\{(\sigma,\sigma(\overleftarrow{x},\overrightarrow{x}))|\sigma\in\Sigma\}$

(3)$E[|e_0+e_1|]=\{(\sigma,(\overleftarrow{n_0},\overrightarrow{n_0}),(\overleftarrow{d_0},\overrightarrow{d_0}))+(\overleftarrow{n_1},\overrightarrow{n_1}),(\overleftarrow{d_1},\overrightarrow{d_1}))|(\sigma,(\overleftarrow{n_0},\overrightarrow{n_0}),(\overleftarrow{d_0},\overrightarrow{d_0}))\in E[|e_0|]\&(\sigma,(\overleftarrow{n_1},\overrightarrow{n_1}),(\overleftarrow{d_1},\overrightarrow{d_1}))\in E[|e_1|]\}$

(4)$E[|e_0-e_1|]=\{(\sigma,(\overleftarrow{n_0},\overrightarrow{n_0}),(\overleftarrow{d_0},\overrightarrow{d_0}))-(\overleftarrow{n_1},\overrightarrow{n_1}),(\overleftarrow{d_1},\overrightarrow{d_1}))|(\sigma,(\overleftarrow{n_0},\overrightarrow{n_0}),(\overleftarrow{d_0},\overrightarrow{d_0}))\in E[|e_0|]$

$\&(\sigma,(\overleftarrow{n_1},\overrightarrow{n_1}),(\overleftarrow{d_1},\overrightarrow{d_1}))\in E[|e_1|]\}$

(5)$E[|e_0\times e_1|]=\{(\sigma,(\overleftarrow{n_0},\overrightarrow{n_0}),(\overleftarrow{d_0},\overrightarrow{d_0}))\times(\overleftarrow{n_1},\overrightarrow{n_1}),(\overleftarrow{d_1},\overrightarrow{d_1}))|(\sigma,(\overleftarrow{n_0},\overrightarrow{n_0}),(\overleftarrow{d_0},\overrightarrow{d_0}))\in E[|e_0|]$

$\&(\sigma,(\overleftarrow{n_1},\overrightarrow{n_1}),(\overleftarrow{d_1},\overrightarrow{d_1}))\in E[|e_1|]\}$

(6)$E[|e_0\div e_1|]=\{(\sigma,(\overleftarrow{n_0},\overrightarrow{n_0}),(\overleftarrow{d_0},\overrightarrow{d_0}))\div(\overleftarrow{n_1},\overrightarrow{n_1}),(\overleftarrow{d_1},\overrightarrow{d_1}))|(\sigma,(\overleftarrow{n_0},\overrightarrow{n_0}),(\overleftarrow{d_0},\overrightarrow{d_0}))\in E[|e_0|]$

$\&(\sigma,(\overleftarrow{n_1},\overrightarrow{n_1}),(\overleftarrow{d_1},\overrightarrow{d_1}))\in E[|e_1|]\}$

第一个公式表示，无论程序处于何种状态 σ，常量(($\overleftarrow{n},\overrightarrow{n}$),($\overleftarrow{d},\overrightarrow{d}$))的值不变，为($\overleftarrow{n},\overrightarrow{n}$),($\overleftarrow{d},\overrightarrow{d}$))。第二个公式表示，变量($\overleftarrow{x},\overrightarrow{x}$)($\overleftarrow{d},\overrightarrow{d}$))在状态 σ 时的值，为其当前的取值 σ($\overleftarrow{x},\overrightarrow{x}$))。第三、四、五、六个公式表示，表达式 e_0 和 e_1 在状态 σ 时和、差、积、商的值分别为 e_0 和 e_1 在状态 σ 时的值的和、差、积、商。

2. 布尔表达式的指称模型

(1)$B[|(\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))|]=\{(\sigma,(\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\}$

(2)$B[|(\overleftarrow{\text{false}},\overrightarrow{\text{false}}),(\overleftarrow{d},\overrightarrow{d}))|]=\{(\sigma,(\overleftarrow{\text{false}},\overrightarrow{\text{false}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\}$

(3)$B[|b_0\wedge b_1|]=\{(\sigma,t_0\wedge_T t_1)|\sigma\in\Sigma\&(\sigma,t_0)\in B[|b_0|]\&(\sigma,t_1)\in B[|b_1|]\}$

(4)$B[|b_0\vee b_1|]=\{(\sigma,t_0\vee_T t_1)|\sigma\in\Sigma\&(\sigma,t_0)\in B[|b_0|]\&(\sigma,t_1)\in B[|b_1|]\}$

(5) $B[|e_0=e_1|]=\{(\sigma,(\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\&A[a_0]\sigma=A[a_1]\sigma\}\cup\{(\sigma,(\overleftarrow{\text{false}},\overrightarrow{\text{false}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\&B[|e_0|]\sigma\neq B[|e_1|]\sigma\}$

(6) $B[|e_0>e_1|]=\{(\sigma,(\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\&A[a_0]\sigma>A[a_1]\sigma\}\cup\{(\sigma,(\overleftarrow{\text{false}},\overrightarrow{\text{false}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\&B[|e_0|]\sigma\leqslant B[|e_1|]\sigma\}$

(7) $B[|e_0<e_1|]=\{(\sigma,(\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\&A[a_0]\sigma<A[a_1]\sigma\}\cup\{(\sigma,(\overleftarrow{\text{false}},\overrightarrow{\text{false}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\&B[|e_0|]\sigma\geqslant B[|e_1|]\sigma\}$

(8) $B[|e_0\geqslant e_1|]=\{(\sigma,(\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))|\sigma\in\Sigma\&A[a_0]\sigma\geqslant A[a_1]\sigma\}$

(9) $B[|e_0\leqslant e_1|]=\{(\sigma,(\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))$

$|\sigma\in\Sigma\&A[a_0]\sigma\leqslant A[a_1]\sigma\}\cup\{(\sigma,((\overleftarrow{\text{false}},\overrightarrow{\text{false}}),(\overleftarrow{d},\overrightarrow{d})))|\sigma\in\Sigma\&B[|e_0|]\sigma>B[|e_1|]\sigma\}$

(10) $B[|e_0\neq e_1|]=\{(\sigma,((\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d})))|\sigma\in\Sigma\&A[a_0]\sigma\neq A[a_1]\sigma\}\cup\{(\sigma,((\overleftarrow{\text{false}},\overrightarrow{\text{false}}),(\overleftarrow{d},\overrightarrow{d})))|\sigma\in\Sigma\&B[|e_0|]\sigma=B[|e_1|]\sigma\}$

3. 语句集的指称模型

对于语句 s 来说，S[|s|]的定义比较复杂。我们首先给定指称为状态之间的关系，接着，用结构归纳法可以看到，每个指称其实是一个部分函数。显然，我们有

(1) $S[skip]=\{(\sigma,\sigma)|\sigma\in\Sigma\}$

(2) $S[abort]=\{(\sigma,\sigma)|\sigma\in\Sigma\}$

(3) $S[|((\overleftarrow{x},\overrightarrow{x})(\overleftarrow{d},\overrightarrow{d})):=e|]=\{(\sigma,\sigma[\frac{n}{(\overleftarrow{x},\overrightarrow{x})}])|\sigma\in\Sigma\&n=E[|e|]\sigma\}$

(4) $S[|s_0;s_1|]=S[|S_0|]\circ S[|S_1|]$，关系的复合即 $S[|s_0;s_1|]=\{(\sigma,\sigma_1)|\sigma\in\Sigma\&\sigma_0\in\Sigma\&\sigma_1\in\Sigma\&(\sigma,\sigma_0)\in S[|s_0|]\&(\sigma_0,\sigma_1)\in S[|s_1|]\}$

(5) $S[|\text{if g fi}|]=\{(\sigma,\sigma')|\sigma\in\Sigma\&\sigma'\in\Sigma\&(\sigma,\sigma')\in G[|g|]\}$

(6) $S[|\text{do g od}|]=\{(\sigma,\sigma')|\sigma\in\Sigma\&\sigma'\in\Sigma\&(\sigma,\sigma')\in G[|g|]\}$

4. 命令的指称模型

同语句的指称模型一样，我们可以用结构归纳法给出监督命令集的指称模型。

(1) $G[|b\rightarrow s|]=\{(\sigma,\sigma')|B[|b|]\sigma=((\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))\&(\sigma,\sigma')\in S[s]\}$

(2) $G[|g_0|g_1|]=\{(\sigma,\sigma_0)|(\sigma,\sigma_0)\in G[|g_0|]\}\cup\{(\sigma,\sigma_1)|(\sigma,\sigma_1)\in G[|g_1|]\}$

第一个公式表示如果监督条件 b 的值为$((\overleftarrow{\text{true}},\overrightarrow{\text{true}}),(\overleftarrow{d},\overrightarrow{d}))\in$DFBool，且在状态 σ 下，语句 s 执行后的状态为 σ'，则当前语言执行后的状态为 σ'。

第二个公式表示任选一个监督语句 g_0 或 g_1 执行，其状态分别为在当前状态 σ 下执行语句 g_0 或 g_1 后的状态。

4 实例分析

从前面对本文所设计的指称语义模型的分析描述中我们知道，动态模糊变量在程序的执行过程中其动态模糊隶属度是会被修改的，修改的原则通常满足 T 模运算[10,11]或 S 模运算，下面我们通过一个简单的实例程序来观察变量的动态模糊隶属度随着程序的执行而发生的修改变化。

DFLexample

{ DFInt $\overrightarrow{x_1}=(\overrightarrow{6},\overrightarrow{0.8}),\overrightarrow{x_2}=(\overrightarrow{22},\overrightarrow{0.8}),\overrightarrow{x_3}=(\overrightarrow{4},\overrightarrow{0.5}),\overrightarrow{x_4}=(\overrightarrow{2},\overrightarrow{0.3})$;

DFReal $\overleftarrow{x_5}=(\overleftarrow{4.5},\overleftarrow{0.9}),\overrightarrow{x_6}$;

DO

$\overrightarrow{x_6}=\overrightarrow{x_2}*\overrightarrow{x_5}-4*\overrightarrow{x_1}*\overrightarrow{x_3}$

IF $\overrightarrow{x_6}>3*\overleftarrow{x_5}\rightarrow\overrightarrow{x_2}=\overrightarrow{x_2}-\overrightarrow{x_4}$

FI

OD

}

运行结果：

由于 T 模运算和 S 模运算有多种形式，这里我们只列举其中的两种形式来观察运行结果。

表 1 程序执行过程

变量	状态 σ_1	状态 σ_2	状态 σ_3
	t(a,b)=min(a,b)		
$\overrightarrow{x_1}$	$(\overrightarrow{6},\overrightarrow{0.8})$	$(\overrightarrow{6},\overrightarrow{0.8})$	$(\overrightarrow{6},\overrightarrow{0.8})$
$\overrightarrow{x_2}$	$(\overrightarrow{22},\overrightarrow{0.8})$	$(\overrightarrow{20},\overrightarrow{0.3})$	$(\overrightarrow{20},\overrightarrow{0.3})$
$\overrightarrow{x_3}$	$(\overrightarrow{4},\overrightarrow{0.5})$	$(\overrightarrow{4},\overrightarrow{0.5})$	$(\overrightarrow{4},\overrightarrow{0.5})$
$\overrightarrow{x_4}$	$(\overrightarrow{2},\overrightarrow{0.3})$	$(\overrightarrow{2},\overrightarrow{0.3})$	$(\overrightarrow{2},\overrightarrow{0.3})$
$\overleftarrow{x_5}$	$(\overleftarrow{5.5},\overleftarrow{0.9})$	$(\overleftarrow{5.5},\overleftarrow{0.9})$	$(\overleftarrow{5.5},\overleftarrow{0.9})$
$\overrightarrow{x_6}$		$(\overrightarrow{25},\overrightarrow{0.5})$	$(\overrightarrow{14},\overrightarrow{0.3})$
	t(a,b)=max(0,a+b−1)		
$\overrightarrow{x_1}$	$(\overrightarrow{6},\overrightarrow{0.8})$	$(\overrightarrow{6},\overrightarrow{0.8})$	$(\overrightarrow{6},\overrightarrow{0.8})$
$\overrightarrow{x_2}$	$(\overrightarrow{22},\overrightarrow{0.8})$	$(\overrightarrow{20},\overrightarrow{0.3})$	$(\overrightarrow{20},\overrightarrow{0.3})$
$\overrightarrow{x_3}$	$(\overrightarrow{4},\overrightarrow{0.5})$	$(\overrightarrow{4},\overrightarrow{0.5})$	$(\overrightarrow{4},\overrightarrow{0.5})$
$\overrightarrow{x_4}$	$(\overrightarrow{2},\overrightarrow{0.3})$	$(\overrightarrow{2},\overrightarrow{0.3})$	$(\overrightarrow{2},\overrightarrow{0.3})$
$\overleftarrow{x_5}$	$(\overleftarrow{5.5},\overleftarrow{0.9})$	$(\overleftarrow{5.5},\overleftarrow{0.9})$	$(\overleftarrow{5.5},\overleftarrow{0.9})$
$\overrightarrow{x_6}$		$(\overrightarrow{25},\overrightarrow{0})$	$(\overrightarrow{14},\overrightarrow{0})$
	$t(a,b)=\frac{a\cdot b}{1+(1-a)(1-b)}$		
$\overrightarrow{x_1}$	$(\overrightarrow{6},\overrightarrow{0.8})$	$(\overrightarrow{6},\overrightarrow{0.8})$	$(\overrightarrow{6},\overrightarrow{0.8})$
$\overrightarrow{x_2}$	$(\overrightarrow{22},\overrightarrow{0.8})$	$(\overrightarrow{20},\overrightarrow{0.21})$	$(\overrightarrow{20},\overrightarrow{0.21})$
$\overrightarrow{x_3}$	$(\overrightarrow{4},\overrightarrow{0.5})$	$(\overrightarrow{4},\overrightarrow{0.5})$	$(\overrightarrow{4},\overrightarrow{0.5})$
$\overrightarrow{x_4}$	$(\overrightarrow{2},\overrightarrow{0.3})$	$(\overrightarrow{2},\overrightarrow{0.3})$	$(\overrightarrow{2},\overrightarrow{0.3})$
$\overleftarrow{x_5}$	$(\overleftarrow{5.5},\overleftarrow{0.9})$	$(\overleftarrow{5.5},\overleftarrow{0.9})$	$(\overleftarrow{5.5},\overleftarrow{0.9})$
$\overrightarrow{x_6}$		$(\overrightarrow{25},\overrightarrow{0.3})$	$(\overrightarrow{14},\overrightarrow{0.21})$

对于上表中变量的动态模糊隶属度变化情况，我们仅以 $t(a,b)=\frac{a\cdot b}{1+(1-a)(1-b)}$ 为例进行解释。首先执行语句 $\overrightarrow{x_6}=\overrightarrow{x_2}*\overrightarrow{x_5}-4*\overrightarrow{x_1}*\overrightarrow{x_3},\overrightarrow{x_2}*\overrightarrow{x_5}$ 运行后的动态模糊隶属度为：

将 $a=0.8,b=0.9$ 代入 $t(a,b)=\frac{a\cdot b}{1+(1-a)(1-b)}=0.3$，故 $\overrightarrow{x_6}=(\overrightarrow{25},\overrightarrow{0.3})$，此时 $\overrightarrow{x_6}>3*\overleftarrow{x_5}$ 的布尔值为真，故 $\overrightarrow{x_2}=\overrightarrow{x_2}-\overrightarrow{x_4}$ 将被执行，此后 $\overrightarrow{x_2}$ 的动态模糊隶属度将会改变，将 $a=0.8$，$b=0.3$ 代入 $\frac{a\cdot b}{1+(1-a)(1-b)}=0.21$，故 $\overrightarrow{x_2}=(\overrightarrow{20},\overrightarrow{0.21})$。程序循环再度执行 $\overrightarrow{x_6}=\overrightarrow{x_2}*\overrightarrow{x_5}-4*\overrightarrow{x_1}*\overrightarrow{x_3}$，同样的方法可得 $\overrightarrow{x_6}=(\overrightarrow{14},\overrightarrow{0.21})$，此时 $\overrightarrow{x_6}>3*\overleftarrow{x_5}$ 已经不能满足，所以程序至此运行结束。

结束语 综上所述，本文的贡献主要有：

(1)根据文献[8]提出的动态模糊程序设计框架，进一步扩充和完善该语言的构造规则。

(2)根据指称语义的原理和方法，通过结构归纳法给出动态模糊逻辑程序设计语言的指称语义模型。

虽然我们已初步通过一个指称语义模型定义了可以处理动态模糊问题的 DFL 程序设计语言，但所做的研究工作还远远不够，以后的工作将围绕动态模糊程序设计语言展开。如将 DFL 的程序框架进一步细化，DFL 程序设计语言的编译与实施等。

参 考 文 献

[1] 屈延文. 形式语义学基础与形式说明. 北京：科学出版社，1989：9-120

[2] 陆汝钤. 计算机语言的形式语义[M]. 北京：科学出版社，1992(12)：95-319

[3] Sebesta R W. 程序设计语言原理[M]. 张勤，译. 北京：机械工业

出版社,2004
[4] Louden K C. 程序设计语言-原理与实践(第二版)[M]. 黄林鹏,毛宏燕,黄晓琴,等译. 北京:电子工业出版社,2004
[5] Partt T W, Zelkowitz M V. 程序设计语言:设计与实现(第四版)[M]. 傅育熙,张冬茉,黄林鹏,译. 北京:电子工业出版,2001
[6] Winskel G. 程序设计语言的形式语义[M]. 宋国新,邵志清,等译. 北京:机械工业出版社,2004(1):45-60
[7] 刘晓杰. 智能神经网络程序设计语言中规则的指称语义[J]. 计算机应用研究,2004(10)
[8] Zhao Xiaofang, Li Fanzhang. The Frame of DFL Programming Language. Journal of Communication And Computer, 3(1):1-15
[9] 李凡长,等. 动态模糊逻辑及其应用. 昆明:云南科技出版社,1997
[10] 李凡长,等. 动态模糊集及其应用. 昆明:云南科技出版社,1997
[11] 李凡长,等. 动态模糊逻辑引论. 昆明:云南科技出版社,2005:8-40
[12] 李凡长,郑家亮. 动态模糊数据的设计语言[J]. 计算机科学,1997(增):54-56
[13] 李凡长,等. 动态模糊数据模型研究. 计算机研究与发展,1998,35(8):714-718
[14] 李凡长. 动态模糊数据运算及模型设计. 计算机工程,2001,27(3):100-102
[15] 李凡长,等. 动态模糊数据模型及其设计语言. 计算机科学,1997,24(12):49-53
[16] Li Fanzhang. Dynamic Fuzzy Logic and Its Applications. Nova Science Publishers, 2007(12)
[17] Zhao Xiaofang, Fan Hui, Liu Xiaohua. Data Types of DFL Programming Language. Fuzzy Systems And Knowledge Discovery. Haikou, Hainan, China, 2007:116-120
[18] Zhao Xiaofang, Li Fanzhang. Denotational Semantics of Dynamic Fuzzy Logic Programming Language // 2006 IEEE International Conference on Granular Computing [C]. Atlanta, USA. May 2006:409-412

(上接第 127 页)

运行 10 次后取平均值得到结果见表 1。

需要指出的是 Libsvm2.82 的内部函数采用 C++语言编写,而这里 PSVRM 与 LSSVMR 都采用 Matlab 语言,但是仍然能看出本文方法的速度优势。从表 1 中可以看出 PSVRM 与 LSSVMR 和 SVR 的拟合能力不相上下,但是学习速度最快。

表 1 PSVRM 与 SVR,LSSVMR 的性能比较

Datasets	PSVRM			SVR(ε=0.01)			LSSVMR		
(train/test/dim.)	MSE	R	Time	MSE	R	Time	MSE	R	Time
Housing(300/206/13)	12.125	0.920	0.016	12.594	0.917	0.218	12.188	0.929	0.048
M-g(800/585/6)	0.015	0.850	0.625	0.017	0.833	2.864	0.014	0.853	1.297
Body-fat(200/52/14)	4.5-e5	0.932	0.015	3.0-e5	0.951	0.165	2.7-e5	0.959	0.088
Space-ga(2000/1107/6)	0.011	0.842	1.406	0.010	0.851	19.62	0.010	0.843	3.156

结束语 本文将用于模式分类的 PSVM 方法,推广到回归估计,给出线性与非线性情形下的 PSVRM 函数估计公式,PSVRM 简单快速,核函数可以不一定满足正定条件。数值实验表明本文方法可行有效,与 SVR 和 LSSVMR 相比,其速度最快,且泛化能力不相上下。下一步的研究工作可基于 PSVRM 在非线性系统辨识、金融时间序列预测、预报建模与控制方面的应用进行。

参 考 文 献

[1] Vapnik V. The Nature of Statistical Learning Theory[M]. New York, Springer, 1995
[2] Drucker H, Burges C J C, Kaufman L. Support vector regression machines. Advances in Neural Information Processing Systems, Cambridge: MIT Press, 1997:155-161
[3] Tay F E H, Cao L. Application of support vector machines in financial time series forecasting[J]. Omega, 2001, 29(4):309-317
[4] Chan W C, Chan C W, Cheung K C, et al. On the modeling of nonlinear dynamic systems using support vector neural networks [J]. Engineering Application of Artificial Intelligence, 2001, 14(2):105-113
[5] 汪西莉,刘芳,焦李成. 基于大规模数据的支撑矢量机的训练和分类[J]. 西安电子科技大学学报,2002,29(1):123-127
[6] Flake G W, Lawrence S. Efficient SVM regression training with SMO[J]. Machine Learning, 2002, 41(1):271-290
[7] 周伟达,张莉,焦李成. 线性规划支撑矢量机[J]. 电子学报,2001,29(11):1-5
[8] Suykens J A K, Vandewalle J. Least Square Support Vector Machine Classifiers[J]. Neural Processing Letters, 1999, 9:293-300
[9] Gestel T V, et al. Financial time series prediction using least squares support vector machines within the evidence framework [J]. IEEE Trans on Neural Networks, 2001, 12(4):809-821
[10] 吴青,刘三阳,杜喆. 回归型模糊最小二乘支持向量机[J]. 西安电子科技大学学报,2007,34(5):773-778
[11] Fung G, Mangasarian O L. Proximal support vector machine classifiers // Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, CA, USA, 2001(8):77-82
[12] Agarwal D K. Shrinkage Estimator Generalization of Proximal Support Vector Machine [C] // Proceeding of the 8th ACM SIGKDD Intl Conference on Knowledge Discovery and Data Mining. Edmonton, Canada, 2002:173-182
[13] Fung G, Mangasarian O L. Multicategory Proximal Support Vector Classifiers[J]. Machine Learning, 2005, 59:77-97
[14] Li K, Huang H. Incremental Learning Proximal Support Vector Machine Classifiers [C] // Proceedings of the 1st International Conference on Machine Learning and Cybernetics. Beijing, 2002, 3:1635-1637
[15] 陶晓燕,姬红兵,马志强. 基于样本分布不平衡的近似支持向量机[J]. 计算机科学,2007,34(5):174-176
[16] Chang C C, Lin C J. LIBSVM-A Library for Support Vector Machines [CP]. 2005. http:// www. csie. ntu. edu. tw/~ cjlin/libsvm/index. html
[17] Murphy P M, Aha D W. UCI repository of machine learning databases [DB]. 2006. http:// www. ics. uci. edu/~ mlearn/ML-Repository. html