The Semantics of Dynamic Fuzzy Logic Programming Language

Xiaofang Zhao

School of Computer Science & Technology, Shandong Institute of Business and Technology
Yantai, China
xf_zh@163.com

Abstract. Dynamic fuzzy problems exist extensively in realistic world. The dynamic fuzzy logic (DFL) programming language is to deal with dynamic fuzzy data. In order to implement DFL programming language, it should be firstly defined. In this paper, we give the denotational semantics of DFL programming language. The work mainly includes modifying the classical lambda calculus to introduce the character of dynamic fuzzy, the descriptions of semantic objects and the handling functions of semantics.

Keywords: Dynamic fuzzy logic (DFL), Programming language, Denotational semantics.

1 Introduction

Events having the character of dynamic fuzzy logic exist extensively. Such as stock market index, economic growth, weather changes and so on. In order to simulate coping with dynamic fuzzy problems in computer, it is necessary to provide software, that is, to research and design a programming language. There have been researches on fuzzy programming language to deal with fuzzy data, such as the references [1,2,3]. And there have been researches on dynamic programming language to deal with dynamic data, such as the reference [4]. Up to now, there has been few programming language to deal with dynamic fuzzy data. In reference [5], we have attempted to give the frame of DFL programming language. In this paper, we define the DFL programming language from the point of denotional semantics. Denotional semantics is based on more mature mathematics theory such as lambda calculus, domains theory and so on. To deal with dynamic fuzzy data, the traditional lambda calculus should been changed.

This paper is organized as follows: In the section 2, the conception of dynamic fuzzy data has been introduced. In section 3, some basic knowledge about lambda calculus has been given. In section 4, the denotational semantics of DFL programming language has been described in detail. In section 5, the conclusion has been presented.

2 Dynamic Fuzzy Data

Pact 1. The character of data with both dynamic and fuzzy is called dynamic fuzzy character.

Eg.1 The global climate becomes more and more warming.

The word "becomes" reflects the dynamic character and "warming" reflects the fuzzy character.

Pact 2. The data with the dynamic fuzzy character is called dynamic fuzzy data.

We call the whole clause mentioned by Eg.1as dynamic fuzzy data.

The sets of dynamic fuzzy datum i.e. dynamic fuzzy sets are defined as follows:

Definition 1. Let a mapping be defined in the domain U.

$(\overleftarrow{A},\overrightarrow{A}):(\overleftarrow{U},\overrightarrow{U})\rightarrow[0,1]\times[\leftarrow,\rightarrow],\ (\overleftarrow{u},\overrightarrow{u})\mapsto(\overleftarrow{A}(\overleftarrow{u}),\overrightarrow{A}(\overrightarrow{u}))$

We write $(\overleftarrow{A},\overrightarrow{A})=\overleftarrow{A}$ or $\overrightarrow{A}$, then we name $(\overleftarrow{A},\overrightarrow{A})$ the dynamic fuzzy sets (DFS) of $(\overleftarrow{U},\overrightarrow{U})$ and name $(\overleftarrow{A}(\overleftarrow{u}),\overrightarrow{A}(\overrightarrow{u}))$ the membership degree of membership function to $(\overleftarrow{A},\overrightarrow{A})$.

Dynamic fuzzy variable is usually symbolized by $(\overleftarrow{x_1},\overrightarrow{x_1}),(\overleftarrow{x_2},\overrightarrow{x_2})$, …or $(\overleftarrow{x},\overrightarrow{x}),(\overleftarrow{y},\overrightarrow{y})$, … where $[\leftarrow,\rightarrow]$ indicates the direction of dynamic change.

3 Lambda Calculus

Lambda calculus is based on function calculus. It is suitable for the description of semantics especially for denotational semantics. With the aim of describing the denotational semantics of dynamic fuzzy logic programming language, we need to modify the classical lambda calculus to introduce the character of dynamic fuzzy. From the definition in the section 2, we can see that to introduce the dynamic fuzzy character we need to add a special set D=[0,1]× [←,→], where "←"denotes good or advance direction of dynamic change and "→"indicates bad or back direction of dynamic change.

To introduce the set D into the classical lambda calculus, a simple method is that terms are represented by trees whose branches labeled with elements of D. In detail, our terms are composed of two parts: one part belongs to the classical lambda calculus and the other part is a list built on the set D. Therefore, for each term M we can write it as [m, l (M)]. Thus term should be redefined.

Definition 2. Terms can be recursively defined as the follows:

If $(\overleftarrow{x},\overrightarrow{x})$ is a variable then for each α∈D, [$(\overleftarrow{x},\overrightarrow{x})$,α] is a term.

If M=[m, *l* (M)] is a term and $(\overleftarrow{x},\overrightarrow{x})$ is a variable then for each α∈D, λ $(\overleftarrow{x},\overrightarrow{x})$.M= [λ $(\overleftarrow{x},\overrightarrow{x})$.m, α·*l* (M)] is a term.

If both M=[m, *l* (M)] and N=[n, *l* (N)] are terms, then for each α∈D, MN= [m n, α·*l* (M) ·*l* (N)] is a term.

Compared with terms of the classical lambda calculus we can see that in our transformed lambda calculus a variable isn't a term any more unless it is labeled by an element of set D.

Below we show an example about terms, see as example Eg.2.

Eg.2 Let m= $\lambda\ \overleftrightarrow{xy}\ .(\overleftarrow{x}\overrightarrow{y}+\overleftarrow{x})\ (\lambda\ \overleftarrow{z}\ .\ \overleftarrow{z}\ )$, where m belongs to the classical lambda calculus part of the term M, and α1,α2, α3, α4, α5, α6, α7, α8, α9, α10∈D which are to label m, then M can be write as:

[$\lambda\ \overleftrightarrow{xy}\ .(\overleftarrow{x}\overrightarrow{y}+\overleftarrow{x})\ (\lambda\ \overleftarrow{z}\ .\ \overleftarrow{z}\ )$,α1·α2·α3·α4·α5·α6·α7·α8·α9·α10]

and its structure as a tree is the following figure Fig.1:

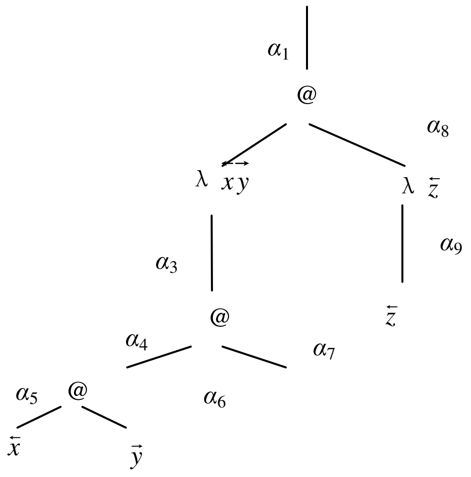

Fig. 1. The structure of Eg.2

Let's analyze above structure: The dynamic fuzzy membership degree of $\overleftarrow{x}$ is α5 and the dynamic fuzzy membership degree of $\overrightarrow{y}$ is α6. While the dynamic fuzzy membership degree of the product of $\overleftarrow{x}$ and $\overrightarrow{y}$ is α4=t (α5, α6) which is different from both α5 and α6 where t is T modular operation [6] or S modular operation [6] of DFS. The rest can be gotten by analogy.

4 The Denotational Semantics of DFL Programming Language

The denotational semantics means making each program element correspond to a mathematical object that called a denotation of its corresponding program element and having each program a mapping from input field to output field where the input field and output field both called domain of discourse. Then we can see that the description of denotational semantics must be based on some mathematics theory such as lambda calculus, domain theory and so on. In general the description of denotational semantics is composed of four parts, namely, the abstract syntax, the conditions of context, the descriptions of semantic objects and the handling functions of semantics. The abstract syntax infers giving the syntactic categories. The conditions of context can example mistakes of syntax. The semantic objects are the mathematical objects mapped from program elements. When describing semantic objects, it means to give the semantic categories. The handling function of semantics can simply be described as mappings from syntactic categories to their corresponding semantic categories. Among the four parts the handling function of semantics is the most important. Below we give the denotational semantics of DFL programming language in detail.

4.1 Abstract Syntax

In this part we give the syntactic structure of the DFL programming language following the guarded commands, see [7], proposed by Dijkstra on account of its characteristic that is to introduce nondeterminism to a program each sentence possibly executed is provided a guarded condition.

S::= skip|
abort|
$(\overleftarrow{v},\overrightarrow{v})$ =E|
(S;S)|
if G fi|
do G od
E::= $(\overleftarrow{v},\overrightarrow{v})$ |
$(\overleftarrow{n},\overrightarrow{n})$ $(\overleftarrow{d},\overrightarrow{d})$ |
E op E
G::=B→S|
(G□G)
B::=(true, $(\overleftarrow{d},\overrightarrow{d})$)|
(false, $(\overleftarrow{d},\overrightarrow{d})$)|
B Bop B|
E rel E
With the following syntactic categories:
S $\in$ sentences
E $\in$ expressions
G $\in$ guarded commands
B $\in$ boolean expressions
$(\overleftarrow{v},\overrightarrow{v})$ $\in$ variables
$(\overleftarrow{n},\overrightarrow{n})$ $\in$ constants
$(\overleftarrow{d},\overrightarrow{d})$ $\in$ D, indicates membership degree
op indicates operation
Bop indicates boolean operation
rel indicates relation operation

4.2 The Conditions of Context

In our case the conditions of context are merged into the handling functions of semantics. When finding mistakes the value is assigned to "error". It doesn't detailedly explain how to deal with mistakes.

4.3 The Descriptions of Semantic Objects

According to the abstract syntax given above, we give the basic syntactic categories of DFL programming language as follows:

Literals :
Positive Numbers Domain Num+ = {0,1,2,...... }
Negative Numbers Domain Num- = {-1,-2,-3,...... }
Integers Domain Int = Num $\oplus$ Num-
Booleans Domain Bool = {True, False}
Dynamic Fuzzy Greeds Domain Grd = [0,1] $\times$ [$\leftarrow$,$\rightarrow$]
Values Domain Val = Int $\oplus$ Bool $\oplus$ Grd
Dynamic Fuzzy Variables Domain DFLVar
Dynamic Fuzzy Constants Domain DFLCon
Dynamic Fuzzy Integers Domain DFLInt
DFLInt=Int $\times$ Grd
Dynamic Fuzzy Booleans Domain DFLBool
DFLBool=Bool $\times$ Grd
Dynamic Fuzzy Values Domain DFLValue
DFLValue=DFLInt $\oplus$ DFLBool
Dynamic Fuzzy Identifiers Domain DFLIde
Dynamic Fuzzy Identifier Values Domain DFLIdeVal
DFLEdeVal=DFLInt
Dynamic Fuzzy Store Domain DFLSto
DFLSto=DFLVar $\rightarrow$ DFLValue
Dynamic Fuzzy Environment Domain DFLEnv
DFLEnv = DFLVar $\rightarrow$ DFLIdeVal

4.4 The Handling Function of Semantics

The basic formation of the handling function of semantics is as follows:

$$E: Exp \rightarrow U \rightarrow S \rightarrow E$$

Where Exp is syntactic category and U→S→E is semantic category. The detailed formation is the following: The syntax is bracketed by the symbol of "[[ ]]" and the semantic parameter is written behind of "[[ ]]".The symbol of "=" is between of them. Below we give the handling functions of semantics of DFL programming language.

S : S→ DFLEnv→DFLSto→S (DFLSto)

$$S\,[[\text{skip}]] = \lambda(\sigma, (\overleftarrow{d}, \overrightarrow{d})).\ (\{\sigma\}, (\overleftarrow{d}, \overrightarrow{d}))$$

$$S\,[[\text{abort}]] = \lambda(\sigma, (\overleftarrow{d}, \overrightarrow{d})).\ (\{\delta\}, (\overleftarrow{d}, \overrightarrow{d}))$$

S [[$(\overleftarrow{v},\overrightarrow{v})$ =E]] =λ(σ, $(\overleftarrow{d},\overrightarrow{d})$).({(DFLSto[[$(\overleftarrow{v},\overrightarrow{v})$]]

(E[[E]] σ)(σ, $(\overleftarrow{d},\overrightarrow{d})$))}, $(\overleftarrow{d},\overrightarrow{d})$)

S [[(S1; S2)]]=ext (S[[S2]] S[[S1]])

S [[if G fi]]= λ(σ, $(\overleftarrow{d},\overrightarrow{d})$).

($(\overleftarrow{d},\overrightarrow{d})$,on2[[λ $(\overleftarrow{x},\overrightarrow{x})$ ∈O.{| δ|}] G [[G]](σ, $(\overleftarrow{d},\overrightarrow{d})$)

S [[do G od]]=fix([λfλ(σ, $(\overleftarrow{d},\overrightarrow{d})$).($(\overleftarrow{d},\overrightarrow{d})$,

on2[λ $(\overleftarrow{x},\overrightarrow{x})$ ∈O.{| σ|}, ext(f)]

(G [[G]](σ, $(\overleftarrow{d},\overrightarrow{d})$)))]

E: Exp→DFLEnv→DFLSto→DFLInt

E[[$(\overleftarrow{v},\overrightarrow{v})$]]=λσ. DFLSto[[$(\overleftarrow{v},\overrightarrow{v})$]]σ

E[[$(\overleftarrow{n},\overrightarrow{n})$ $(\overleftarrow{d},\overrightarrow{d})$]]=λσ.[$(\overleftarrow{n},\overrightarrow{n})$ $(\overleftarrow{d},\overrightarrow{d})$]

E[[E1 op E2]]= λσ.let[$(\overleftarrow{n_1},\overrightarrow{n_1})$ $(\overleftarrow{d_1},\overrightarrow{d_1})$]

=E[[E1]]σ in let[$(\overleftarrow{n_2},\overrightarrow{n_2})$ $(\overleftarrow{d_2},\overrightarrow{d_2})$]

=E[[E2]]σ in [$(\overleftarrow{n_1},\overrightarrow{n_1})$ op $(\overleftarrow{n_2},\overrightarrow{n_2})$, s($(\overleftarrow{d_1},\overrightarrow{d_1})$, $(\overleftarrow{d_2},\overrightarrow{d_2})$)]

G :G→DFLEnv→DFLSto→DFLBool

G [[B→S]]= λσ.B [[B]]σ

G [[G1□G2]]= λσ.(G [[G1]]σ∧G [[G2]]σ)

B : BExpr→DFLEnv→DFLSto→DFLBool

B [[$(\overleftarrow{true},\overrightarrow{true})(\overleftarrow{d},\overrightarrow{d})$]]=λσ.[$(\overleftarrow{true},\overrightarrow{true})(\overleftarrow{d},\overrightarrow{d})$]

B [[$(\overleftarrow{false},\overrightarrow{false})(\overleftarrow{d},\overrightarrow{d})$]]=λσ.[$(\overleftarrow{false},\overrightarrow{false})(\overleftarrow{d},\overrightarrow{d})$]

B [[B1 Bop B2]]= λσ.let[b1, $(\overleftarrow{d_1},\overrightarrow{d_1})$]= B [[B1]]σ in let[b2, $(\overleftarrow{d_2},\overrightarrow{d_2})$]

= B [[B2]]σ in [$(\overleftarrow{n_1},\overrightarrow{n_1})$ Bop $(\overleftarrow{n_2},\overrightarrow{n_2})$, s($(\overleftarrow{d_1},\overrightarrow{d_1})$, $(\overleftarrow{d_2},\overrightarrow{d_2})$)]

Where s is T modular operation[6] or S modular operation[6] of DFS.

5 Example

Below we show a simple program to observe the process of dealing with dynamic fuzzy data.

DFLexample

{DFInt $\overrightarrow{x_1} = (\vec{6},\overrightarrow{0.7})$, $\overrightarrow{x_2} = (\overrightarrow{12},\overrightarrow{0.8})$;

DFInt $\overrightarrow{x_3} = (\vec{4},\overrightarrow{0.6})$, $\overrightarrow{x_4} = (\vec{2},\overrightarrow{0.5})$;

DFReal $\overleftarrow{x_5} = (\overleftarrow{2.5},\overleftarrow{0.7})$, $\overrightarrow{x_6}$;

DO

\\ SQRT()denotes computing square root

$\overrightarrow{x_6}$ =SQRT($\overrightarrow{x_2}$ * $\overrightarrow{x_2}$ −4* $\overrightarrow{x_1}$ * $\overrightarrow{x_3}$)

```
    IF $\overrightarrow{x_6} > \overleftarrow{x_5}$
                    Then    $\overrightarrow{x_2} = \overrightarrow{x_2} \bullet \overrightarrow{x_4}$
   FI
  OD
}
```

Results:
From the denotational semantics above given we see the value of dynamic fuzzy membership degree will be changed through T modular operation or S modular operation during the execution of the program. There are several forms of T modular and S modular operation, but here we only take example for two kinds of them.

Table 1. The result of DFLexample

Name of	stateσ1	stateσ2	stateσ3
t(a , b)=min(a , b)			
$\overrightarrow{x_1}$	$(\vec{6},\overrightarrow{0.7})$	$(\vec{6},\overrightarrow{0.7})$	$(\vec{6},\overrightarrow{0.7})$
$\overrightarrow{x_2}$	$(\overrightarrow{12},\overrightarrow{0.8})$	$(\overrightarrow{10},\overrightarrow{0.5})$	$(\overrightarrow{10},\overrightarrow{0.5})$
$\overrightarrow{x_3}$	$(\vec{4},\overrightarrow{0.6})$	$(\vec{4},\overrightarrow{0.6})$	$(\vec{4},\overrightarrow{0.6})$
$\overrightarrow{x_4}$	$(\vec{2},\overrightarrow{0.5})$	$(\vec{2},\overrightarrow{0.5})$	$(\vec{2},\overrightarrow{0.5})$
$\overleftarrow{x_5}$	$(\overleftarrow{2.5},\overleftarrow{0.7})$	$(\overleftarrow{2.5},\overleftarrow{0.6})$	$(\overleftarrow{2.5},\overleftarrow{0.6})$
$\overrightarrow{x_6}$		$(\overrightarrow{6.9},\overrightarrow{0.6})$	$(\vec{2},\overrightarrow{0.5})$
t(a , b)=max (0,a+ b-1)			
$\overrightarrow{x_1}$	$(\vec{6},\overrightarrow{0.7})$	$(\vec{6},\overrightarrow{0.7})$	$(\vec{6},\overrightarrow{0.7})$
$\overrightarrow{x_2}$	$(\overrightarrow{12},\overrightarrow{0.8})$	$(\overrightarrow{10},\overrightarrow{0.3})$	$(\overrightarrow{10},\overrightarrow{0.3})$
$\overrightarrow{x_3}$	$(\vec{4},\overrightarrow{0.6})$	$(\vec{4},\overrightarrow{0.6})$	$(\vec{4},\overrightarrow{0.6})$
$\overrightarrow{x_4}$	$(\vec{2},\overrightarrow{0.5})$	$(\vec{2},\overrightarrow{0.5})$	$(\vec{2},\overrightarrow{0.5})$
$\overleftarrow{x_5}$	$(\overleftarrow{2.5},\overleftarrow{0.7})$	$(\overleftarrow{2.5},\overleftarrow{0.7})$	$(\overleftarrow{2.5},\overleftarrow{0.7})$
$\overrightarrow{x_6}$		$(\overrightarrow{6.9},\vec{0})$	$(\vec{2},\vec{0}.)$

6 Conclusion

Compared to relevant work, what is the characteristic in this paper is that we combined the character of dynamic with the character of fuzzy for research but not only the character of fuzzy as the reference [3] or only the character dynamic as reference [4] and described the semantics of DFL programming language in the terms of denotational semantics.What we have done in this paper is summarized as follows:

We have modified the classical lambda calculus to introduce a special set D whose elements will be used to label its terms.

We have gavine denotational semantics of DFL programming language which include abstract syntax, the descriptions of semantic objects and the handling functions of semantics.

References

1. Adamo, J.M.: L.P.L. A Fuzzy Programming Language: 1. syntactic aspects. J. Fuzzy Set and Systems 3, 151–179 (1980)
2. Adamo, J.M.: L.P.L. A Fuzzy Programming Language: 2. Semantic Aspects. J. Fuzzy Set and Systems 3, 261–289 (1980)
3. Alvarez, D.S., Gó mez Skarmeta, A.F.: A fuzzy language. J. Fuzzy Sets and System 141, 335–390 (2004)
4. Tang, Z.-s.: Temporal Logic programming and software engineering (in Chinese). Science Press, Beijing (2002)
5. Zhao, X.: The Frame of DFL Programming Language. In: 7th International Conference on Fuzzy Systems And Knowledge Discovery, Hai Nan, China, pp. 343–348 (2010)
6. Li, F.-z., Zheng, J.-l.: Module Operation of Dynamic Fuzzy Stes. Journal of the Central University for Nationalities 16, 96–101 (1997) (in Chinese)
7. Lu, R.-q.: Formal semantics of computer languages. Science Press, Beijing (1992) (in Chinese)