関数型言語Validへのオブジェクト指向的記述形式の導入とその実現方式

7F−3 渡辺直一郎 鶴田直之 日下部茂 谷口倫一郎 雨宮真人
九州大学 大学院 総合理工学研究科

1 まえがき

筆者らは、大規模化・複雑化するニューラルネットワークの研究を支援するための超並列実行環境及びプログラミング環境の実現を目指している。そして、並列処理計算機の基礎方式としてデータフロー方式が有効であると考え、Datarolアーキテクチャ及びAMPアーキテクチャを提案している[1][2]。また、プログラミング言語としては、データフロー計算機用関数型言語Valid[3]を用いている。

これまで、筆者らは、自律的に動作するニューロンの計算モデル(非履歴依存タイプ及び履歴依存タイプ)及びネットワークの自律的な高並列計算モデルをValidにより記述している[4]。さらに、ニューロンを構成する各種機能関数のカプセル化、ネットワークを構成するニューロン(オブジェクト)と各種機能関数のカプセル化及びニューロンやネットワークの仕様の抽象化が可能となることによるプログラミング性能向上、効率的なプログラムの再利用・メンテナンスを実現するために、Validにオブジェクト指向的な記述形式を導入している[4][5]。

本稿では、オブジェクト指向的記述形式を加えたValidにより記述されたプログラムを、Validにより記述されたプログラムに変換する方式について述べる。Validに導入したオブジェクト指向的記述形式は、高階関数の概念に基づいたものであり、変換されたプログラムもValidによる高階関数の記述を含んだものとなる。筆者らが提案しているDatarolマシンは、高階関数の実行が可能であり[6]、変換されたプログラムの実行環境となり得る。

2 オブジェクト指向的記述形式

本節では、Validに導入したオブジェクト指向的記述形式の概要を述べる。詳細は参考文献[4][5]を参照されたい。

【クラス定義】

```
class ClassName(SystemParameterPart):SuperClassName
= {
      [メンバオブジェクト定義部]
      [メンバ値定義部]
      [メンバ関数定義部]
  }
```

SystemParameterPartには、システムパラメータを型指定して列挙する。親クラスが存在する場合は、そのクラス名をSuperClassNameに指定する。ただし、親クラスは、1つしか指定できない。なお、システムパラメータとは、ニューロンやネットワークの仕様を決定するようなパラメータ(例えば、ネットワークを構成するニューロン数、ニューロンにおけるシグモイド関数の傾きなどの各種係数など)のことである。メンバオブジェクト定義部には、当該クラス(ネットワーク)を構成するオブジェクト(ニューロン)を定義できる。外部から参照され得るメンバ関数定義は、public指定しておく。

【継承】

子クラスは、親クラスのシステムパラメータ及び各メンバ定義をすべて継承する。また、子クラスにおいて、各メンバの再定義を許す。その場合、その子孫クラスに継承されるメンバ定義は、再定義されたメンバ定義となる。

【オブジェクトの生成及び定義】

システムパラメータに実引数を与えることにより、その仕様を満たすクラスオブジェクトが生成される。生成されたオブジェクトは、値定義と同じ要領で、オブジェクト定義によりオブジェクト名を付与される。

```
ObjectName:object = ClassName(ActualPar1,...)
ObjectName:array(1,object)
   = foreach (i) in (0..9) do ClassName(ActualPar1,...)
```

【メンバ関数の参照】

オブジェクト名を用いて、そのオブジェクトのメンバ関数を参照できる。ただし、public指定されているメンバ関数に限る。

```
ObjectName.MemberFuncName(ActualPar1,...)
ObjectName[i].MemberFuncName(ActualPar1,...)
```

3 プログラムの変換

上で述べたオブジェクト指向記述形式を加えたValidにより記述されたプログラムは、Validにより記述されたプログラム(関数定義と関数参照により記述されたプログラム)に変換される。Validに導入しているオブジェクト指向的記述形式は、高階関数の概念に基づいており、変換されたプログラムも高階関数の記述を含んだプログラムとなる。

List1にオブジェクト指向的記述形式を加えたValidにより記述された例題プログラムを、List2にList1のプログラムから変換されたプログラムを示す。List1及びList2を参考にして、変換の方式の概要について述べる。

変換では、メインの関数と処理内で実際に参照される関数やメンバの定義のみが変換されて出力される。(クラスC3は参照されていないので、変換されない。)

子クラスにおいて継承されるメンバ定義は、そのクラスから先祖関係の近いクラスで定義されているメンバ定義となる。(例えば、クラスC3に継承されているメンバ関数Func1の定義は、クラスC2におけるメンバ関数Func1の定義である。)

生成されたオブジェクトのクラスC4及びC2は、システムパラメータと参照されるメンバ関数名を仮引数とする関数定義に変換される。(List2中のC4(sysp:int,fname:string)、C2(sysp1,sysp2:int,fname:string)に相当する。) この関数は、fnameに応じた関数の部分適用(List2中のC4()の

Implementation of Object−Oriented Programming Feature into Functional Programming Language Valid

Naoichiro WATANABE, Naoyuki TSURUTA, Shigeru KUSAKABE, Rin-ichiro TANIGUCHI, Makoto AMAMIYA
Interdisciplinary Graduate School of Engineering Sciences, Kyushu University

C4_Func4(sysp)、C2() の C2_Func2(sysp1,sysp2) に相当する。) を返す関数である。

生成されたオブジェクトのクラス C4 及び C2 内のメンバオブジェクト定義 (クラス C4 内ののobj) は、そのクラス名を付加した一意な名前の関数定義に変換される。(List2 中の C4_obj() に相当する。) その際、システムパラメータを関数の仮引数に加える。(C4_obj() における C4 の sysp が相当する。) この関数は、関数の部分適用 (List2 中の main() の obj の C4(2)、C4_obj() の C2(2,2) と C2(4,4) に相当する。) を返す関数である。

関数定義本体内のオブジェクト定義は、型指定での object を function に変更する。(List2 中の main() の obj:**function** に相当する。)

生成されたオブジェクトのクラス C4 及び C2 内のメンバ値定義、メンバ関数定義 (継承されているメンバ定義も含む) は、それぞれ、そのクラス名を付加した一意な名前の関数定義に変換される。(List2 中の C4_Func4()、C2_Func2()、C2_Func1()、C2_Val1() に相当する。) その際、システムパラメータを関数の仮引数に加える。(C4_Func4() におけるクラス C4 の sysp、C2_Func2() におけるクラス C2 の sysp1 及び sysp2、C2_Func1() 及び C2_Val1() における C1 の sysp1 が相当する。)

なお、List2 の main() における obj("Func4")(input) は、関数 C4() の部分適用 (obj に相当する。) に残りの引数 (文字列"Func4" に相当する。) を与え、その結果返される関数 C4_Func4() の部分適用に残りの引数 (input に相当する。) を与えていることを示している。List2 の C4_Func4() における obj[i](“Func2”)(input[i]) についても同様である。

4 まとめ

本稿では、オブジェクト指向的記述形式を加えた Valid により記述されたプログラムを、Valid により記述されたプログラムに変換する方式について述べた。Valid に導入しているオブジェクト指向的記述形式は、高階関数の概念に基づいたものであり、よって、無理なく Valid により記述されたプログラムへの変換を実現できる。変換されたプログラムは高階関数の記述を含むプログラムとなる。

参考文献

[1]M.Amamiya,R.Taniguchi:“Datarol : A Massively Parallel Architecture for Functional Langage”,proc. 2th IEEE Symposium on Parallel and Distributed Processing,pp726-735(1990.12)

[2]R.Taniguchi,M.Amamiya:“AMP : An Autonomous MultiProcessor for Image Processing and Computer Vision”,Proc. 10th ICPR,vol.2 pp497-500,IEEE Computer Society Press(1990.6)

[3] 長谷川隆三, 雨宮真人:“データフローマシン用関数型高級言語 valid”, 電子情報通信学会論文誌,vol.J71-D,No.8,pp1532-1539(1988.8)

[4] 鶴田直之, 谷口倫一郎, 雨宮真人:“オブジェクト指向によるニューラルネットワークの記述”, 第 3 回自律分散システム・シンポジウム資料,pp113-118(1992.1)

[5] 渡辺直一郎, 鶴田直之, 谷口倫一郎, 雨宮真人:“超並列実行環境下におけるニューラルネットワークのプログラミング”, 電子情報通信学会技術研究報告,Vol.92,No.230,pp1-8(1992.9)

[6] 河内久和, 高橋英一, 谷口倫一郎, 雨宮真人:”Datarol マシンへの高階関数及び遅延評価の実現”, 情報処理学会第 44 回全国大会講演論文集 (6),pp119-120(1992.3)

List1. オブジェクト指向的記述形式を加えた Valid により記述された例題プログラム

```
class C1(sysp1:int)
{
  val1:int = 2*sysp1;

  public
  function Func1(par:int) return(int)
  = val*sysp1*par;
};

class C2(sysp2:int):C1
{
  public
  function Func2(par:int) return(int)
  = { let a:int=sysp2*par in Func1(a) };

  function Func1(par:int) return(int)
  = (sysp1+sysp2)*par;
};

class C3(sysp3:int):C2
{
  public
  function Func3(par:int) return(int)
  = sysp3+par;
}

class C4(sysp:int)
{
  obj:array(1,object)
    = foreach (i) in (0..sysp−1) do
        if (i<sysp/2) then C2(2,2) else C2(4,4);

  public
  function Func4(input:array(1,int)) return(array(1,int))
  = foreach (i) in (0..sysp−1) do obj[i].Func2(input[i]);
};

function main() return(array(1,int))
= { let obj:object=C4(3),
        input:array(1,int)=(4,3,5)
    in  obj.Func4(input) };
```

List2. List1 のプログラムから変換されたプログラム

```
function main() return(array(1,int))
= { let obj:function = C4(sysp)
        input:array(1,int)=(4,3,5)
    in  obj(“Func4”)(input) };

function C4(sysp:int,fname:string) return(function)
= case fname==“Func4” −> C4_Func4(sysp) end;

function C4_obj(sysp:int) return(array(1,function))
= foreach (i) in (0..sysp−1) do
   if (i<sysp/2) then C2(2,2) else C2(4,4);

function C4_Func4(sysp:int,input:array(1,int))
return(array(1,int))
= { let obj:array(1,function) = C4_obj(sysp)
    in  foreach (i) in (0..sysp−1) do
          obj[i](“Func2”)(input[i]);

function C2(sysp1,sysp2:int,fname:string) return(function)
= case fname==“Func2” −> C2_Func2(sysp1,sysp2) end;

function C2_Func2(sysp1,sysp2,par:int) return(int)
= { let a:int=sysp2+par in C2_Func1(sysp1,sysp2,a) };

function C2_Func1(sysp1,sysp2,par:int) return(int)
= (sysp1+sysp2)*par;

function C2_val1(sysp1:int) return(int)
= 2*sysp1;
```