第 33 卷 第 4 期 计 算 机 工 程 2007 年 2 月
Vol.33 No.4 Computer Engineering February 2007
·软件技术与数据库· 文章编号：1000—3428(2007)04—0032—02 文献标识码：A 中图分类号：TP31

智能神经网络程序设计语言的研究与实现

黄雪梅[1,2]，李 涛[1]，徐春林[1]，卢 暾[1]
(1. 四川大学计算机学院，成都 610065；2. 四川大学电气信息学院，成都 610065)

摘 要：将神经网络与逻辑推理统一到面向对象理论中，建立了同时具备神经网络和专家系统特性、融合连接机制和符号机制的智能神经元模型。提出了一种新的程序设计语言——智能神经网络程序语言(NIPL)，实现了神经计算、逻辑推理和数值计算的统一。定义了 NIPL 的语法，设计并实现了智能神经网络程序设计语言 NIPL 编译器，从而为开发智能神经网络应用系统提供了有效的手段。

关键词：智能神经元模型；智能神经网络程序设计语言；抽象语法；NIPL 编译器

Research and Implementation of Intelligent Neural Network Programming Language

HUANG Xuemei[1,2], LI Tao[1], XU Chunlin[1], LU Tun[1]
(1. School of Computer, Sichuan Univ., Chengdu 610065; 2. School of Electric & Information, Sichuan Univ., Chengdu 610065)

【Abstract】 As ideas of neural networks and logic inference are introduced into the theory of object-oriented, an intelligent neuron model is built. This model, with the characteristic of neural networks and expert system, integrates the link mechanism with the symbol mechanism. Then a new programming language——intelligent neural network programming language called NIPL is presented. NIPL makes the unification of neural network computation, logic inference and numerical computation. The abstract grammar of NIPL is given. The compiler of NIPL is designed and implemented and thus an effective means for development of the intelligent system is provided.

【Key words】 Intelligent neuron model; Intelligent neural network programming language (NIPL); Abstract grammar; NIPL compiler

基于符号逻辑的传统人工智能(AI)系统研究和基于示例学习的人工神经网络(NN)研究是研究人类智能的两大基本途径，前者较擅长于推理理解工作，而后者更适合于完成信息感知的功能，二者在功能上互补且都具有自身的局限性。智能神经网络系统理论[1]把AI与NN有机结合，使它们各自的优势得到充分利用。

该理论将面向对象的思想与神经网络理论结合，抽象出了大神经元概念[2]，实现了面向神经元的程序设计语言(ONPL)[3]。将法则、事实以及神经计算等概念统一到面向对象的理论中，提出了一种新的人工智能程序设计语言ROOP[4]，实现了数值计算与符号推理的统一。综合ONPL和ROOP，提出了智能神经元模型以及智能神经网络程序设计语言(NIPL)[5]。

1 智能神经元模型

将面向对象的思想引入到神经网络中，对传统的神经元及神经网络概念扩展，抽象出大神经元，在此基础上引入符号逻辑理论，使其既能按神经网络的方法获取及表达知识，又能将已有的知识以逻辑规则的形式纳入智能神经元对象中，并且使用符号逻辑进行推理。于是得到智能神经元的一个更具一般意义的模型——智能神经元模型。如图 1 所示。

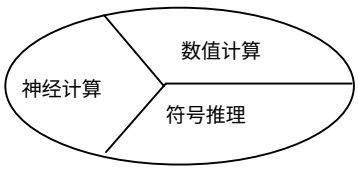

图 1 智能神经元模型

如同人脑分工一样，图 1 中的智能神经元模型同时具备模糊计算、精确的数值计算以及符号推理的功能，3 种机制并发执行。神经元对象的属性和行为大体上分为 3 类：

(1)与神经计算有关的属性及行为；

(2)与逻辑推理有关的属性及行为；

(3)与传统数值计算有关的属性及行为。

2 智能神经网络程序设计语言

NIPL 将智能神经元模型描述为智能神经元类，智能神经元则是智能神经元类的实例。对于程序员来讲，智能神经元是一种新的高级抽象数据结构，数据和运算方法(或行为、或属性，如神经计算、符号推理等)被封装成一个整体——智能神经元。通过对象继承机制，一个智能神经元类可以继承已有的智能神经元类，可以加入新的属性及行为，也可以重新定义有关的行为。

NIPL语言在语法上承袭了ONPL语言[3]以及ROOP语言[4]，对神经网络的描述采用ONPL的描述手段，而逻辑推理则使用ROOP的描述方式加以表达。下面详细定义NIPL语言的抽象语法，NIPL语言的指称语义参见文献[6]。

(1)NIPLProgram $\triangleq$ nal:Neuron_definition;
vdl:Variable_Declaration_list;
body:NiplStmt;

基金项目：国家自然科学基金资助项目(60373110)；教育部博士点基金资助项目(20030610003)

作者简介：黄雪梅(1966－)，女，副教授、博士生，主研方向：人工智能，网络安全；李 涛，教授、博导；徐春林、卢 暾，博士生

收稿日期：2006-03-11 **E-mail：**hxm-scu@sina.com

(2) Neuron_definition ≜ sNeuron_definition｜
iNeuron_definition;

(3) Variable_Declaration_list ≜ Variable_declaration;

(4)Variable_Declaration ≜ Neuron_Variable_Declaration｜ Cpp_Variable_Declaration;

(5)Neuron_Variable_Declaration ≜ t:Neuron_type；
v:Identifier;

(6)Cpp_Variable_Declaration ≜ t:Cpp_type; v:Identifier;

(7)Cpp_type ≜ Simple_type｜ Point_type｜ Array_type

(8) Type ≜ Cpp_type｜ Neuron_type；

(9)Neuron_type ≜ “neuron”｜ “iNeuron”｜ Identifier；

(10)sNeuron_definition ≜
class_name: Type_name;
public_parent: Parent_List;
public_attribute: Variable_declaration_list;
public_behavior: Function_declaration_list;

(11)iNeuron_definition ≜
class_name: Type_name; //类名
public_parent: Parent_List; //父类类名表
old: Layer_definition; //输出层定义
hldl: Layer_definition;//隐含层定义
ild: Layer_definition; //输入层定义
con: Connection; //网络拓扑定义
public_attribute: Variable_declaration_list;
public_behavior: Function_declaration_list;
rules:ROOPProgam; //逻辑规则

(12)ROOPProgam ≜ Workstorages:Class_declaration_lis;
Rules: Rule_declaration_list;

(13)Rule_declaration_list ≜ Rule_Declaration;

(14)Rule_Declaration ≜ name:Identifier;
variables:Variable_Declaration_list;
Left:LHS;Right:RHS;

(15)LHS ≜ lcr:Predicate;templates:Templates_list;
filter:Filter;

(16)Predicate ≜ id:Identifier;arguments:Express_list;

(17)Template_list ≜ Patterens;

(18)Patterens ≜ Simple_Patterens｜ Compound_Patteren;

(19)Simple_Patterens ≜ name:WorkStorage_Name;
pttr:Express_list;

(20)Compound_Patterens ≜ Var:Variable;
name:WorkStorageName;
Patterens:Express_list;

(21)Filter ≜ Express;

(22)RHS ≜ variables:Variable_declaration_list;
actions:Action;

(23)Actions ≜ General_Command:Command｜
lcs:Predicate;

(24)Layer_definition ≜ Layer_element;

(25)Layer_element ≜ Varible_declaration｜ Variable;

(26) Connection ≜ Connection_element;

(27)Connection_element ≜ LinkStmt｜ DeleteLinkStmt;

(28)NiplStmt ≜ LinkStmt｜ DeleteLinkStmt｜ LoadStmt｜ SaveStmt｜ InputStmt｜ OutputStmt｜ CppStmt

(29)LinkStmt ≜ na:Variable;nb:Variable;

(30)DeleteLinkStmt ≜ na:Variable;nb:Variable;

(31)LoadStmt ≜ no:Variable;file:File_variable;

(32)SaveStmt ≜ no:Variable;file:File_variable;

(33)InputStmt ≜ no:Variable;data:Variable;

(34)OutputStmt ≜ no:Variable;data:Variable;

(35)File_variable ≜ Identifier;

其中，(10)定义了简单神经元类。(11)定义了智能神经元类，是智能神经元模型在 NIPL 中的体现，智能神经元相当于一个特殊对象，其输入层、中间层、输出层、网络拓扑、成员变量相当于该对象的属性，而成员函数则相当于该对象的行为。神经元及神经网络的原始定义可以完成神经计算；成员函数可用于数值计算；逻辑规则与推理机相结合可进行符号逻辑推理。(12)~(23)描述 NIPL 的逻辑规则。WorkStorage 表示事实数据库，是一个具有属性和行为的对象。lcr、lcs 分别是规则间逻辑通信的接收谓词和发送谓词。Command 包括条件、循环、函数调用等。逻辑规则可看作是一类特殊的对象成员，其右部 *RHS* 中的活动可是智能神经元的成员函数，如此，逻辑规则不仅可完成逻辑推理，而且可作用于智能神经元的学习、计算和推理。(24)~(25)是层定义。(26)~(27)网络拓扑声明。(28)~(35)描述 NIPL 命令。

3 NIPL 的实现

本文以下详细讨论 NIPL 语言的单机实现，有关 NIPL 语言的分布式实现参见文献[7]。

3.1 NIPL 实现方式

NIPL 是通过预编译方式来实现的。NIPL 程序的处理流程如图 2 所示。

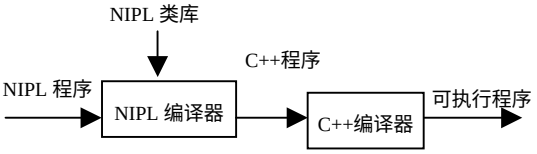

图 2　NIPL 程序处理流程

3.2 NIPL 编译器的实现

NIPL 编译器的结构如图 3 所示。

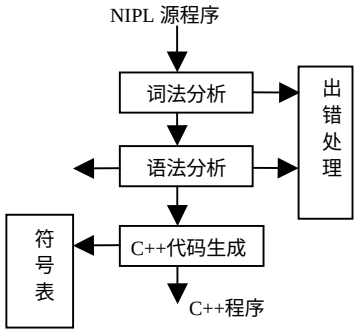

图 3 NIPL 编译器总体结构

NIPL 源程序的处理过程：(1)生成源文件的行信息(用于报告错误位置)，记录所有被双引号扩起来的字符串的起始及结束位置，并将所有注释用空白符代替；(2)进行词法、语法分析，建立符号表；(3)根据符号表生成 C++源程序及相应的 Make 文件；(4)使用 Visual C++的 NMAKE 程序及编译连接程序生成可执行程序。

(1)词法分析。NIPL 词法单位集合是 C/C++词法单位集合的超集。在词法分析器中定义了一个枚举类型 TOKEN，对所有的关键字、运算符、分隔符、注释标记以及其它词法单

(下转第 36 页)