DOI:10.3969/j.issn.1001-4551.2013.06.030

基于构件的梯形图算法模块封装方法*

吴盼盼,严 义*
(杭州电子科技大学 计算机学院, 浙江 杭州 310018)

摘要:针对传统的梯形图无法实现通用算法的封装复用及算法保密性问题,首先,从基于构件技术的梯形图实现原理出发,对构件的属性定义进行了详细地分析;其次,通过封装技术的继承性、可复用性以及基于梯形图数据结构模式的构件语言重写,设计出了一种将智能算法封装成梯形图中可复用构件模块的方法;最后,以整型权值的神经网络为例,介绍了在梯形图算法生成平台上将C语言编辑的神经网络测试算法封装成固定的构件模块的方法,并与实现相同功能的传统梯形图编程方法进行了比较。研究结果表明,该方法适用于任何常用算法的封装,并且由程序员自行进行智能算法模块的封装,大大减少了软件开发活动中大量的重复性工作。另外,封装技术的保密性同时也保证了内部算法的安全性。

关键词:梯形图; 构件; 复用; 梯形图; 神经网络
中图分类号:TP311 **文献标志码**:A **文章编号**:1001-4551(2013)06-0764-05

Encapsulation of algorithm module based on component technology in ladder diagram programming

WU Pan-pan, YAN Yi
(College of Computer, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: Aiming that the traditional ladder diagram programming can't realize generic algorithm multiplexing and algorithm secrecy problem, firstly, the component technology based on the ladder diagram principle was analyzed from the attribute and the definition in detail. Then, through the inheritance and reusability of the encapsulation technology and the rewrite of the component language which based on the ladder diagram data structure model, a method of packaging the intelligent algorithm into the reusable component module in ladder diagram was designed. At last , the type of the weights of the neural network was taken as an example, a method of packaging the neural network test algorithm which was edited by C language into a fixed component module was introduced, and it was compared with the method that realized the same function with the traditional ladder diagram programming. The results indicate that, this method is suitable for the encapsulation of any commonly algorithms, and packaging the intelligent algorithm module by the programmer themself can reduce a lot of repetitive work in software development activities. The confidentiality of the packaging technology ensures the security of the internal algorithm.

Key words: ladder diagram; component; reuse; ladder chart; neural network

0 引 言

构件封装与复用技术[1]可把构件的属性和操作结合在一起,组成一个独立的对象,之后可对其进行重复使用。目前,封装与复用技术的研究主要集中在面向对象程序设计的软件领域。如李志伟、赵建平等[2-3]提出了面向对象技术中关于类的封装与复用技术;张荣等[4]提出了关于组件及动态库的封装及复用;深晴霓、杜虹等[5]提出了一种基于完整性度量架构的数据封装方法。然而,这些文献中提出的封装方法均无法

收稿日期:2013-01-09
基金项目:国家自然科学基金资助项目(61272189)
作者简介:吴盼盼(1988-),女,浙江温岭人,主要从事智能与控制方面的研究. E-mail:wupanpan987@163.com
通信联系人:严 义,男,教授,硕士生导师. E-mail:yybjyyj@163.com

在工业控制领域得到广泛的应用。在工业控制领域，关于封装与复用技术的使用研究很少，虽然 Reza Ezuan Samin 等[6]提出了适合开发 PLC 程序的可重用自动化组件，但对于这些组件的研究实现均为静态封装方式，即每个封装好的构件只能实现基本的操作功能，并且编程者只能使用具有特定功能的基本构件进行编程。

然而仅使用基本指令构件编辑的梯形图程序存在可读性差、实现数字算法复杂等缺点，故国内外学者对于在嵌入式 PLC 的图编程中嵌入其他语言进行混合编程也做了不少研究。例如西门子编程平台采用了在梯形图中嵌入结构化文本（ST）进行混合编程；基恩士的软件平台能够在梯形图中嵌入“类”C 脚本语言进行混合编程。然而对于结构化文本或者是脚本语言的嵌入采用的均是文本框的形式，未经过封装，这使得程序失去了整体的可读性及美观性，也无法实现算法的保密性，而且对于类似西门子系统的开发软件，若想使用结构化文本，还需另外安装 SCL 软件包，这使得软件平台的安装也变得复杂。此外，若采用静态封装的方法将上述嵌入的程序进行封装，则封装后的构件功能过于单一，无法灵活地应用在不同的环境中。因此，若能采用动态的封装方式将算法进行封装，则能够大大提高编程效率。

尽管部分学者已经在动态封装方式上做了研究，例如将多条 IL 指令进行组合并封装成一个新的模块，再将其添加到梯形图编程平台中，以实现较为复杂的功能，然而采用 IL 指令进行编程对于复杂算法的实现较困难，也增加了代码量。为此，本研究基于构件技术提出一种算法构件模块封装的方法，使用梯形图算法生成平台中的构件标准，并结合梯形图中嵌入高级语言的方法，最后运用神经网络的测试过程对该方法进行验证。

1　算法构件模型

1.1　构件定义及构件模型

1.1.1　定义 1：构件模型

构件模型[7]除了定义构件的本质属性外，还向外界提供其他构件调用的入口服务，同时构件请求外界服务引用结构。故构件可描述为 Component<C-Properties，C-Services，C-References，Implementation>，其中：

（1）C-Properties 表示构件的属性集，包含了构件的各种属性信息；

（2）C-Services 表示构件提供的服务集，包含了该构件为其他关联构件提供的服务信息；

（3）C-References 表示构件需要的引用集，包含了其他相关构件为其提供的各种服务信息；

（4）Implementation 表示构件本身的实现，对构件本身的实现方法与功能进行了描述。

1.1.2　定义 2：构件接口类型

构件的接口根据其用途可定义为两类：逻辑关系接口和数据接口。其中，逻辑关系接口包括串联接口和并联接口两种；数据接口包括输入数据接口、输出数据接口以及辅助数据接口 3 种。

传统构件模型的接口数量固定，即输入、输出的接口数量固定，且固定模块的功能名一定，设计者无法更改，只能使用构件库中固有的功能块进行程序设计。

1.2　算法构件模型

根据定义 1 及定义 2，算法构件模型可采用传统的功能模块构件实现，然而，由于自定义函数的参数个数是动态的，若采用传统的静态构件模型，即构件的功能名固定，且对应构件模型的输入、输出端口固定，即输入参数及输出参数的个数也将固定，而不能动态地设置构件的功能名及输入参数与输出参数的个数，这使得构件框架变得复杂多样，用户难以正确地选择和使用。此外，构件库中的功能块有限，用户只能使用已经存在的构件模块进行编程，这对于设计者造成了很大的限制，也降低了编程效率。

因此，采用动态构件模型，可依赖于任何功能构件模块，也无数据接口个数的限制，可支持用户自己封装所需的功能构件模块，并可支持数据接口数量多样化需求。算法构件模型的外形采用只含逻辑关系接口而不含实际的数据输入、输出接口的形式，在构件属性中，用户可以动态地定义函数的参数（输入、输出均可通过形参的方式实现）。

基于原有构件模型的基础，本研究将算法构件模块的外形设计成与原 PLC 算法生成平台中已有构件相同。算法构件模型如图 1 所示。

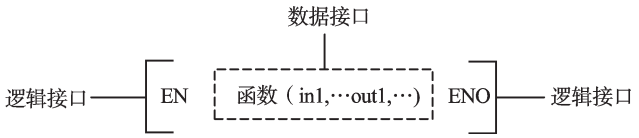


图 1　算法构件模型

根据定义 1 所述的统一构件的表达形式，算法构件模型可描述为 LDC<Name，ID，Properties，Services，References，Implementation>。基于构件标准，算法构件也包含基本构件的常用属性，构件属性如表 1 所示。

1.3　算法构件的 XML 保存形式

从数据描述语言的角度看，XML 语言具有灵活、

表1 构件属性

属性名	说明
ControlSize	显示比例
Location	位置
Size	大小
ILRow	对应IL指令号
Comment	注释
State	该指令当前状态值
Content	内容
RightUp	向上并联竖线
RightDown	向下并联竖线

可扩展、有良好的结构和约束,有助于理解和代码的自动生成等优点,此外,采用XML形式来保存构件信息在异构领域具有良好的兼容性。

算法构件的XML保存形式如下:

```
<Object type="构件存储路径,构件库,Version=版本号,Culture=neutral,PublicKeyToken=null" name="构件名称">
  <Property>
    <ControlSize>构件单位大小</ControlSize>
    <Location>构件位置</Location>
    <Size>构件大小</Size>
    <ILRow>指令号</ILRow>
    <Content>函数名</Content>
    <Comment>注释</Comment>
    <State>当前状态</State>
  </Property>
</Object>
```

2 可复用算法构件模块的封装方法

封装[8]是指可构件的属性和操作结合在一起,组成一个独立的对象。其内部信息对外是隐藏的,用户只能看到对象封装界面上的信息;外界只能通过接口与对象发生联系。软件复用[9]则可在软件开发过程中避免重复劳动,提高软件开发的效率。本研究采用构件形式将算法封装起来,通过外部接口的连接访问实现构件复用,该技术具有现实意义。

2.1 算法构件模块的封装技术

算法构件封装模块结构体中的变量主要用来描述被封装模块的属性,如函数名、函数参数、函数体及函数功能说明。

该构件封装模块结构体代码如下:

```
public struct FUNInformation
{
    public string FUNName;// 函数名
    public string FUNPara;// 函数参数
    public ArrayList FUNContent;// 函数体
    public string FUNInstruction;// 函数功能说明
}
```

算法构件封装模块采用统一管理,提供的操作有添加、修改、删除等。

算法构件模块封装流程示意图如图2所示。

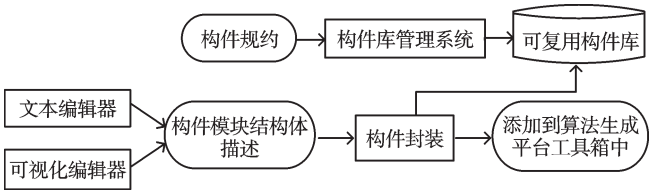


图2 算法构件模块封装流程示意

(1) 封装并添加算法模块。用户通过文本编辑器或者可视化编辑器编辑完自定义函数后,输入需要添加的函数名及参数等,系统将在编辑文档中查询与指定函数名符合的相关信息(例如函数形参、函数体等),并提取该函数的全部信息,根据构件模块的结构体描述将相关信息添加到公共列表中,最后将封装好的构件模块加载到工具箱中的自定义函数模块的树形结构下,并在可复用构件库中添加新构件的描述信息。

(2) 修改已封装的算法构件模块。即将算法的全部信息进行更新,用户修改算法构件模块时,同样需在编辑文档中查找与指定模块函数名相同的算法信息,同时对其进行函数名、参数个数等检错分析,若需要更新的信息准确无误,则修改公共列表中的对应模块的信息并可复用构件库中的构件信息与工具箱中的信息;否则将相应的错误提取出提醒开发人员,并保持原构件模块不进行修改。在添加或修改功能块时,可能会出现函数重名或者输入参数个数不正确的可能,这些错误信息均会一一列出以提醒开发者。

(3) 删除已封装的算法构件模块。用户在执行删除操作时,只需输入需要删除的算法模块的函数名,通过指定的函数名得到相应模块的全部信息,然后从公共列表中删除该模块的全部信息,并且删除工具箱中对应的算法构件模块及分复用构件库中的相应描述信息。若取消则关闭封装界面,不进行任何操作。

2.2 算法管理方式

为了保证管理上的统一性,算法构件封装模块的算法信息亦采用XML语言进行描述集保存,保存文件命名为PfuncModes.xml,且该文件以加密方式进行保存。

下面给出封装模块的基本框架示例:

```
<PfuncModes>
  <Object>
    <name>函数名</name>
    <Para>函数形参</Para>
    <Content>函数体</Content>
```

```
    <Instruction>函数功能说明</Instruction>
  </Object>
</PfuncModes>
```

2.3　算法处理方式

对于封装好的算法构件模块，其处理过程主要分为两大步骤：

（1）算法构件模型的编译。封装好的算法构件模型可按原来的梯形图编译方法进行编译：首先对梯形图程序进行串并连接分析，通过拓扑抽象将其转化为有向图；再根据串联/并联规约，生成二叉分解树；最后根据IL语法库，执行语义翻译算法，形成IL指令[10]，经宏处理后加载到应用程序工程中，作为调用内部算法的入口点。

（2）内部算法的处理。算法构件模块中的参数处理方式类似于C语言的参数处理过程。通过检查接口输入、输出参数的正确性；其次将内部算法中的形参替换成接口传递进入的实参；再读取梯形图内部地址映射表，根据内部地址映射表，将算法中涉及的相关地址转化为嵌入式硬件系统中对应的逻辑存储地址；最后将完整的算法添加到C语言编译器中进行编译[11]。

3　实验结果与分析

本研究以整型权值的神经网络[12]为例，对如何将用C语言实现的神经网络测试算法封装成固定模块，并且在原有梯形图编程平台上使用的方法做了介绍，最后与实现相同功能的传统梯形图编程方法进行比较。

神经网络主要分为学习与测试两部分，该实验的学习部分在Matlab中实现，学习算法采用了“可再生的差分进化算法”[13]，学习结束后，记录所得权值；测试部分在梯形图中实现，即在梯形图中设置与训练时神经网络相同的参数，完成构建后，导入权值，进行测试。此处以“I-H-O”表示神经网络的拓扑结构（神经网络结构如图3所示），根据该拓扑结构，该实验构建了一个“2-3-1”的简单神经网络作为示例进行说明，即输入神经元个数为2，隐层神经元个数为3，输出层神经元个数为1。

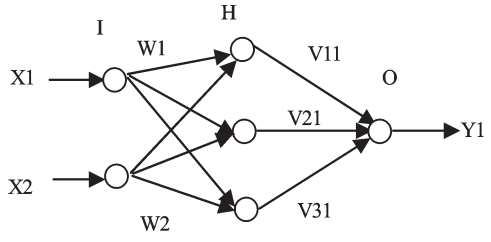
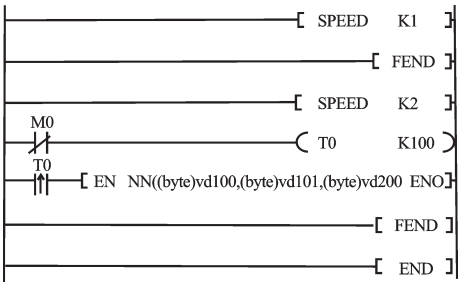


图3　神经网络结构

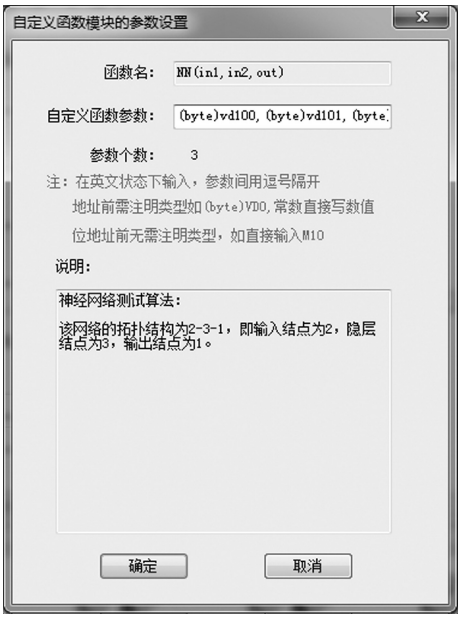
I—输入层神经元个数；H—隐层神经元个数；O—输出层神经元个数

3.1　梯形图内嵌封装后的神经网络算法实现

神经网络测试算法封装模块的梯形图调用如图4所示，梯形图逻辑过程控制中，当常闭触点M0为逻辑0时，定时器T0产生1 s脉冲，T0每秒获得一个上升沿，执行一次神经网络测试算法。该算法模块是经过封装过后的构件模块，用户在编辑梯形图时可以直接对其进行调用，而无需知道其内部算法如何实现，采样值通过地址VD100、VD101传入，输出则存储于VD200中。



（a）梯形图调用



（b）参数设置对话框

图4　神经网络测试算法封装模块的梯形图调用

内部算法实现过程如下：

（1）定义神经网络结构，将之前在Matlab中学习所得的权值传入相应的数组iIn2HidWeight，iHid2OutWeight中，并将输入采样值传入输入数组中；

（2）根据输入层到隐层的权值，计算每个隐层结点的累加和，并根据传输函数确定隐层的输出；

（3）根据隐层到输出层的权值，计算输出结点的累加和，并根据传输函数确定输出结果，并将结果保存到地址vd200中。

3.2　神经网络算法的梯形图实现

基于梯形图算法生成平台，本研究采用传统梯形

图实现相同测试功能的方法如图5所示。先将之前在Matlab中学习所得的权值传入地址vd110~vd120中，之后的实现方法与内部算法实现步骤相同。

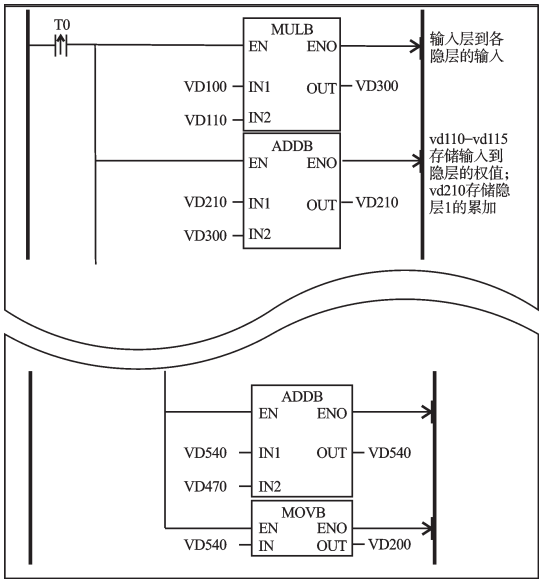


图5 梯形图实现的神经网络测试算法

定时器T0每秒获得一个上升沿，执行一次神经网络测试算法。由于神经网络的测试过程是由许多运算过程组成的，故整个梯形图代码主要使用了加法指令(ADDD)、减法指令(SUBD)、乘法指令(MULD)、除法指令(DIVD)、数据传送指令(MOVD)、数据比较指令(CMPD)。此外，整个梯形图中使用到的地址范围为vd100~vd540，这说明为了存储各个神经元的输入输出，需开辟大量的地址来存储输入输出的值。

3.3 两种实现方法的比较分析

本研究从梯形图大小、程序可读性及保密性等方面对上述两种实现方法进行比较，得出的结果如表2所示。由表2可以看出，调用封装后的神经网络算法模块较梯形图设计方式不仅大大缩短了代码量，而且优化了程序的可读性，此外，这种方式还很好地保护了内部算法。

表2 两种算法实现方式的比较

性能 / 实现方式	梯形图大小/KB	数据内存空间/B	程序可读性	保密性
梯形图设计	149	288	冗长、难懂	不保密
封装算法模块实现	12	48	精简、易读	算法保密

因此，通过将广泛使用的各种智能算法封装好并添加到梯形图算法生成平台上，不仅大大减少了软件开发活动中大量的重复性工作，提高了软件生产率，缩短了开发周期，提高了软件的灵活性和标准化程度，还大大提高了算法的保密性。

4 结束语

本研究以基于梯形图算法生成平台为基础，在原来实现梯形图与C语言混合编程的基础上，提出了一种将自定义算法封装成专用模块的方法，这便于在不同领域进行复用。并以简单的神经网络算法为例，证明了基于该方法，对于算法的编辑及新构件的制作与使用，开发者和使用者无需再学习额外的规范语言和工具，提高了系统的可靠性、容错性及安全性。

在下一步的研究中，研究者可以整合各种常用的智能算法封装模块，在梯形图中建立智能算法构件库，方便工业控制的使用。

参考文献(References)：

[1] 史浩辉，何 炜，基于构件的指控软件复用[J]. 计算机技术与发展，2011，21(2)：159-161.

[2] 李志伟. VB环境下基于类的软件复用技术研究[J]. 计算机工程与设计，2010，31(5)：1152-1155.

[3] 赵建平，赵建辉，顾 培，等. 一种基于数据库和面向对象的软件复用技术[J]. 兵工自动化，2011，30(8)：92-96.

[4] 张 荣，聂 飞，黄海莹，等. 测控软件算法的组件封装技术研究[J]. 电子设计工程，2011，19(8)：22-25.

[5] 深晴霓，杜 虹，文 汉，等. 一种基于完整性度量架构的数据封装方法[J]. 计算机研究与发展，2012，49(1)：210-216.

[6] SAMIN R E，LEE M J，ZAWAWI M A. PID Implementation of Heating Tank in Mini Automation Plant using Programmable Logic Controller(PLC)[C]//International Conference on Electrical，Control and Computer Engineering. Pahang，Malaysia：[s.n.]，2011：21-22.

[7] 王 伟. 面向PLC控制算法的构件模型研究[D]. 杭州：杭州电子科技大学计算机学院，2010：24-30.

[8] ROBINSON R M，WARD P A S. An Architecture for Reliable Encapsulation Endpoints using Commodity Hardware [C]//2011 30th IEEE International Symposium on Reliable Distributed Systems. Portugal：[s.n.]，2011：4-7.

[9] 彭波兰. 基于特定领域的软件复用研究[J]. 应用技术，2011，24(1)：173-174.

[10] YAN Yi，ZHANG Hang-ping. Compiling Ladder Diagram into Instruction List to comply with IEC 61131-3[J]. **Computers in Industry**，2010，61(5)：448-462.

[11] 黄小强，严 义，邬惠峰，等. PLC梯形图中内嵌C语言编程的实现[J]. 机电工程，2012，29(4)：421-424.

[12] BAO Jian，ZHOU Bin. Optimization of Neural Network with Fixed-point Weights and Touch-screen，Calibration [C]// ICIEA 2009. 4th IEEE Conference. Xi'an：[s.n.]，2009：3704-3708.

[13] BAO Jian，CHEN Yu，YU Jin-shou. A Regeneratable-Dynamic Differential Evolution Algorithm for Neural Networks with Integer Weights[J]. **Journal of Zhejinag Univ-Sci C (Comput & Electron)**，2011，11(12)：939-947.

[编辑：李 辉]