# 基于 AOV 图存储 PLC 梯形图的方法

张惠杰，林伟敏
（福州大学 数学与计算机科学学院，福建 福州 350000）

**摘 要**：提出一种直接以 AOV（Activity On Vertex）图存储 PLC（Programmable Logic Controller）梯形图的方法。编辑梯形图的同时，修改 AOV 图，然后根据 AOV 图的拓扑结构更新梯形图图符坐标，最后进行绘制显示。该方法无需进行梯形图向 AOV 图的转换，通过操作规则的约束来替代语法的检查，使梯形图的编辑更加便捷和规范。详细介绍了 AOV 图的编辑过程和坐标的更新算法。对 AOV 图向二叉树的转换算法进行修改，使其能适应于所有 AOV 图，并给出了相应的实例。

**关键词**：可编程逻辑控制器；梯形图；AOV 图；指令表
**中图分类号**：TM571　　**文献标识码**：A　　**文章编号**：1674-7720(2012)16-0070-04

## PLC ladder diagram editing method based on AOV diagraph and conversion algorithm to instruction list

Zhang Huijie, Lin Weimin
(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350000, China)

**Abstract**: A PLC ladder diagram (LD) editing method using AOV diagraph to restore LD directly is proposed. AOV diagraph will be changed when the LD is being modified. Then updates LD symbol coordinates according to the structure of the AOV diagraph and display the LD to user. The method does not need a conversion from LD to the AOV diagraph and alternates syntax check by the constraints of the operating rules, which makes LD editing more convenient and norms. The AOV diagraph's editing process and coordinates update algorithm are described in detail. A modified conversion algorithm from AOV diagraph to binary tree is presented. The algorithm can adapt to all kinds of AOV diagraph, and a corresponding instance is given.

**Key words**: PLC; ladder diagram; AOV diagraph; instruction list

梯形图是使用最多的图形编辑语言，被称为 PLC 的第一编程语言。梯形图以图符的形式直观地再现了各逻辑控件的电器连接关系，并用串、并联等拓扑关系组织图符的顺序位置来表述逻辑。梯形图形象直观，但对于 PLC 来说是不可执行代码，无法直接运行，需事先转换成指令表。指令表是一系列符合 IEC61131-3 标准的指令的集合。对嵌入式 PLC 系统来说，研究梯形图向语句表的转换算法及其实现技术是必要的。PLC 梯形图转换为指令表通常包括 5 个步骤[1]。参考文献[1-3]对梯形图存储结构、语法检查的规则做了详细介绍。但对梯形图的编辑没有限制，可任意绘制，从而导致处理复杂、语法检查规则繁琐。因此本文提出了直接以 AOV 图对梯形图进行存储的方法，编辑梯形图的同时，进行相应的规则约束，动态生成 AOV 图。该过程将梯形图编辑、语法检查和 AOV 图的生成同时完成，使常用的 5 个步骤缩短为 3 个，如图 1 所示。该方法与常用方法相比更为简便、快捷。

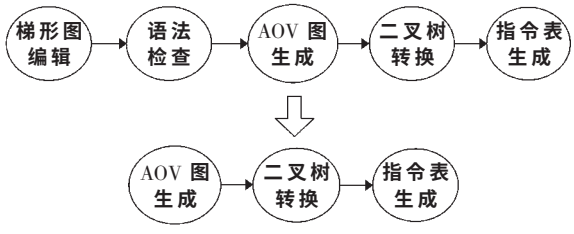



图 1　PLC 指令表生成过程

## 1 AOV 图及其数据结构

AOV 图是一种用顶点表示活动，用弧<$i$,$j$>表示活动 $i$ 必须在活动 $j$ 之前完成的有向图，其中 $i$ 称为 $j$ 的前驱，$j$ 称为 $i$ 的后继。

PLC 的梯形图程序由若干图符按一定的规则链接而成，其自上而下、自左向右的执行方式本质上就是一种 AOV 图，因此本文直接将梯形图中的图符以 AOV 图的结构进行存储，其中横线不存储，竖线存储为虚节点。如图 2 中上图为梯形图，下图为梯形图在内存中实际的存储结构。AOV 图中普通图符有行和列两个坐标值，如 X8(2,4)表示 X8 在梯形图中第 2 列第 4 行。虚节点有 3 个坐标值，分别表示虚节点的列坐标、行起始坐标和行结束坐标，如 V3(2,1,3)表示该虚节点在第 2 列，起始位置为第 1 行，结束位置为第 3 行，文中规定虚节点列坐标的取值为其左边相邻位置的列坐标。

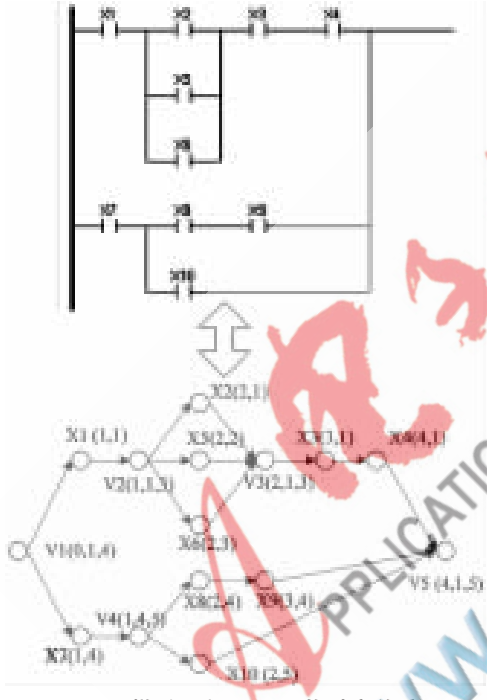


图 2 梯形图和 AOV 图的对应关系

所有顶点使用一个链表进行存储，访问时对该链表进行遍历。

## 2 坐标的更新

按照以上的对应关系，对梯形图进行修改时，其实也就是对 AOV 图进行修改。对梯形图的修改操作有很多：插入串联节点、插入并联节点、删除串联节点、删除并联节点、插入并联分支、插入输出分支等，如果对各种操作进行分析，根据插入、删除的各种不同情况更新 AOV 各个顶点的坐标，处理复杂、繁琐。因此本文提出一种直接通过 AOV 图拓扑结构生成 AOV 图各个顶点坐标的算法。该算法只需对修改后的 AOV 图重新进行坐标的生成，而无需理会具体的操作。算法的具体流程如下：

(1)申请一个存放 AOV 顶点的指针堆栈、当前列坐标 *CurrentX*、当前行坐标 *CurrentY*、临时变量 $x_1$、$y_1$、$x_2$、$y_2$，AOV 顶点指针为 $P_1$、$P_2$、$P_3$，并将 $P_1$ 指向 AOV 图中入度为 0 的顶点。*CurrentX* 初始化为 0，*CurrentY* 初始化为 1；

(2)循环直到 $P_1$ 指向的节点的第一个后继节点的列坐标为 11(文中描述的系统只提供 11 列的标记，最后一列固定为输出节点或功能块)，循环过程中 $P_1$ 指向的节点如果为虚节点，则虚节点的列坐标设置为 *CurrentX*，更新标记置位，若虚节点出度大于 1，则将虚节点指针压入堆栈中，$P_1$ 指向虚节点的第一个出度；若 $P_1$ 指向的节点为图符节点，则 *CurrentX* 加 1，将该节点的列坐标设置为 *CurrentX*，行坐标设置为 *CurrentY*，更新标记置位，$P_1$ 指向图符节点的后继节点。

(3)从堆栈中取出栈顶指针赋给 $P_1$，循环直至堆栈为空。循环过程中进行如下操作：①初始化变量：*CurrentX* 设为 $P_1$ 指向虚节点的列坐标，$P_2$、$P_3$ 取为 $P_1$ 所指向的虚节点的第一个坐标未更新的后继节点，若该虚节点除了 $P_2$ 指向的节点外仍有未更新的后继节点，则将该虚节点的指针再次压入堆栈。②获取 *CurrentY* 的值：$x_1$ 设为 $P_1$ 指向虚节点的列坐标，$y_1$ 设为 $P_1$ 指向虚节点的最后一个已更新过坐标的后继节点的行坐标。如图 3 中，若 $P_1$ 指向 V1，$P_2$ 指向 X8，则 $x_1=0$，$y_1=1$。做如下循环操作：循环直至 $P_2$ 指向的节点为虚节点，且虚节点的第一个后继的行坐标小于等于 $y_1$ 时停止；循环中 $P_2$ 赋值为其指向节点的第一个后继节点。循环结束后，将 $x_2$ 设为 $P_2$ 指向虚节点的列坐标。仍以 X8 为例，最后 $P_2$ 指向 V5 时停止，$x_2=5$。至此获得$(x_1,x_2)$组成的区间。遍历该区间内已更新过坐标的节点，并获取这些节点中行坐标的最大值，并将 *CurrentY* 设为该最大值加 1。③更新该行直至 $P_2$ 指向虚节点之间的节点坐标：此时 $P_3$ 指向 $P_1$ 所指向的虚节点的第一个坐标未更新的后继节点，循环直至 $P_3$ 指向的节点为 $P_2$ 指向的虚节点。循环过程中分以下几种情况进行处理：①若 $P_3$ 指向节点为图符节点且其后继也为图符节点，则 *CurrentX*+1，将该节点的列坐标设

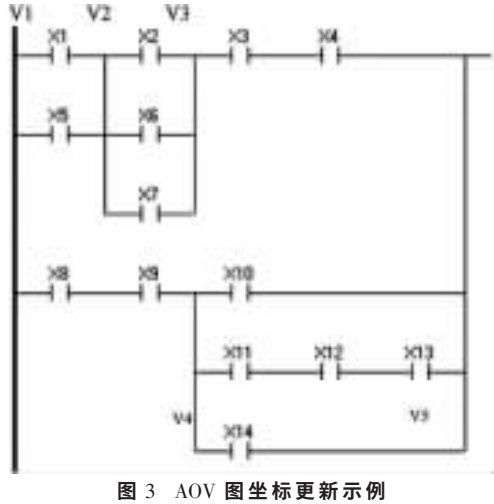


图 3 AOV 图坐标更新示例

置为 *CurrentX*，行坐标设置为 *CurrentY*，更新标记置位，$P_3$ 指向图符节点的后继节点；②若 $P_3$ 指向节点为图符节点，且其后继为与 $P_3$ 指向节点列坐标相同的虚节点，说明已更新好坐标的节点中需要插入一个新节点，某些坐标需要向右移动一个位置。此时需遍历 AOV 图的存储链表，寻找列坐标大于等于 $P_3$ 指向节点列坐标的所有虚节点，将其列坐标加 1，并依次寻找这些虚节点的后继节点，沿着这些后继节点向右遍历，直至虚节点停止，将找到的后继节点列坐标加 1；③若 $P_3$ 指向节点为虚节点，则虚节点的列坐标设置为 *CurrentX*，更新标记置位，若虚节点出度大于 1，则将虚节点指针压入堆栈中，$P_3$ 指向虚节点的第一个出度。

## 3 AOV 图的编辑

不是所有的操作对 AOV 图都是有效和正确的，因此首先对 AOV 图的编辑进行相应的规则约束，以保证 AOV 图拥有正确的拓扑结构，使最后生成的梯形图符合标准，无语法错误。

对梯形图进行一些全局约束：梯形图中只提供 11 列元件编辑位置，最后一列固定为输出线圈或功能块。每个网络建立之后默认生成一个输出线圈，参数待用户修改。因为每个网络都必须有一个输出，其输出为输出线圈或功能块。

对 AOV 图（或称梯形图）的编辑只提供以下操作：添加串联开关、添加并联开关、添加输出分支、删除节点，能基本满足编辑的需要。

## 4 指令表的生成

在完成了对 AOV 图的编辑后，需要将 AOV 图转换成二叉树，再对二叉树进行后续遍历即可获得指令表。参考文献[4-6]提出的各种基于二叉树的 AOV 图转换为指令表算法都无法适用于本文中的 AOV 图，参考文献[7]给出的方法无法适应于虚节点出度或入度大于 2 的情况。本文对参考文献[7]提出的算法进行了修改，使其能适用于各种 AOV 图向二叉树的转换。修改后的算法流程如图 4 所示。

(1)创建两个二叉树节点指针堆栈——“与堆栈”和“或堆栈”，分别用于保存二叉树中的“与”和“或”节点指针，并初始化两个堆栈为空。二叉树初始时为一个根节点 Root，无左右子树。申请图符顶点指针 $P_1$ 和二叉树顶点指针 $P_2$，并将 $P_1$ 指向 AOV 图中入度为 0 的顶点，$P_2$ 指向 Root。

(2)从“与堆栈”中弹出“与”节点指针，并赋给 $P_2$。

(3)创建一个“与”节点，并赋给 $P_3$，如果 $P_2$ 的左子树为空，则将 $P_3$ 指向节点作为 $P_2$ 的左子树，否则将 $P_3$ 指向的节点作为 $P_2$ 的右子树。$P_1$ 所指向的节点作为 $P_3$ 的左子树。$P_2$ 新指向 $P_3$。

新建一个临时 AOV 图顶点指针堆栈 stack，申请一个临时顶点指针 tp，从 $P_1$ 所指顶点的第二个后继开始，

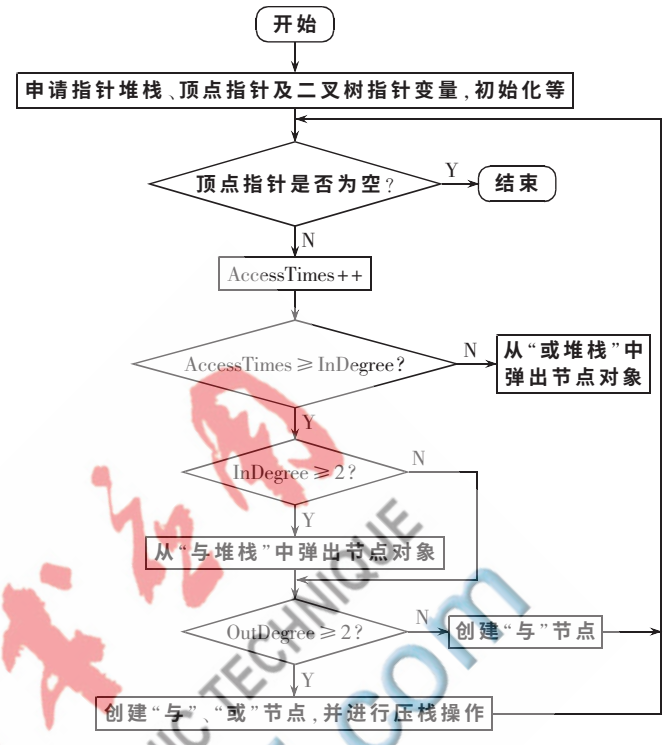



图 4 AOV 图转换为二叉树流程图

做如下操作，直至最后一个后继：设当前为第 *OutNum* 个后继，将 tp 指向该后继。tp 沿其第一个后继寻找起始行坐标小于等于 $P_2$ 的第 *OutNum*−1 个后继行坐标的虚节点，找到后停止。找到时如果 stack 为空，则将 $P_1$ 的第 *OutNum* 个后继和 tp 压入 stack；当 stack 不为空时，如果 tp 指向的虚节点列坐标大于等于 stack 的堆顶的指针所指向的虚节点的列坐标，则将 $P_1$ 的第 *OutNum* 个出度和 tp 压入 stack。

至此得到 $P_1$ 指向虚节点后继的执行顺序，越靠近 stack 堆顶的顶点越后执行。根据该 stack 建立“与”和“或”节点：从 stack 弹出一个图符顶点和一个虚节点，分别赋给 NP 和 VP。新建一个“或”节点，将其赋给 $P_4$。在“与堆栈”中查找 VP：①若未找到，则新建一个“与”节点，将其赋给 $P_3$，并将 $P_3$ 和 VP 压入“与堆栈”中。若 $P_2$ 的左子树为空，则将 $P_3$ 作为 $P_2$ 的左子树，否则将 $P_3$ 作为 $P_2$ 的右子树。$P_4$ 作为 $P_3$ 的左子树。将 $P_4$ 和 NP 压入“或堆栈”中，并将 NP 的压栈标志 stacked 置位，将 $P_2$ 赋给 $P_4$。②若在“与堆栈”中找到 VP，则不新建“与”节点，若 $P_2$ 的左子树为空，则将 $P_4$ 作为 $P_2$ 的左子树，否则将 $P_4$ 作为 $P_2$ 的右子树，并将 $P_4$ 和 NP 压入“或堆栈”中，将 NP 的压栈标志 stacked 置位，将 $P_2$ 赋给 $P_4$。

循环以上操作，直至 stack 为空。再将 $P_1$ 新指向当前 $P_1$ 所指顶点的第一个出度。

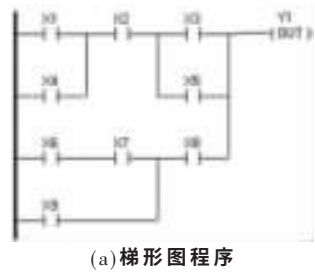
(4)从“或堆栈”中弹出图符顶点对象赋给 $P_1$，再弹出二叉树节点对象赋给 $P_2$；查找沿 $P_1$ 的支路上未建立“与”、“或”节点的后继，并为其建立“与”、“或”节点。

将 $P_1$ 的前驱赋给 VP，计算 $P_1$ 在 VP 后继中的序号，记为 *OutNum*。接下来的过程与步骤(3)类似，只是操作从VP 的第 *OutNum*+1 个后继开始，直至第一个已入栈的后继结束，且 $P_1$ 也不指向顶点的第一个出度。这里不再详述。

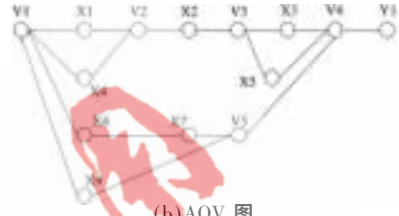
(5)创建一个“与”节点，并将该节点赋给 $P_3$；若 $P_2$ 节点的左子树为空，则将 $P_3$ 指向的节点作为 $P_2$ 的左子树，否则将 $P_3$ 指向的节点作为 $P_2$ 的右子树；然后将 $P_1$ 指向的节点作为 $P_3$ 的左子树，并使 $P_2$ 指向 $P_3$ 对应的节点，$P_1$ 新指向当前 $P_1$ 所指顶点的后继。因为出度小于 2，所以只有一个或没有后继。

图 5 给出了该算法的具体实例。图 5(a)为显示时所看到的梯形图程序，图 5(b)为内存中实际的存储结构，图 5(c)为按上述算法生成的二叉树。将图 5(c)中二叉树的输出节点及其父节点去除，再去除二叉树中多余的与节点和虚节点则得到图 5(d)中的二叉树，对其进行后序遍历即可得到图 5(e)中的指令表。

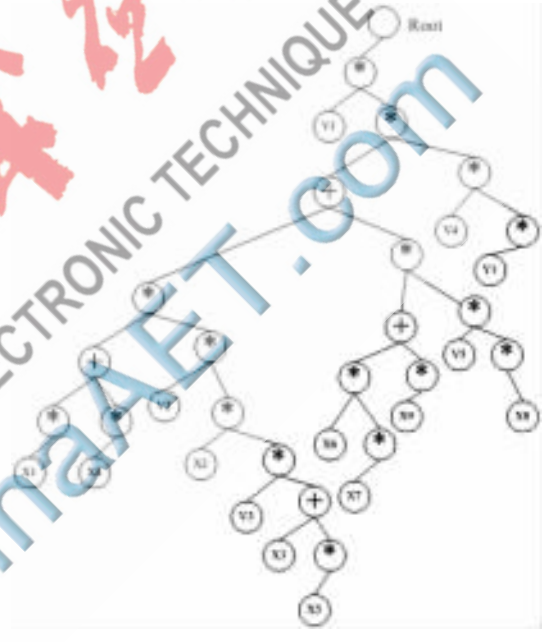
本文提出了一种直接编辑 AOV 图的方法来编辑 PLC 梯形图，以 AOV 图的数据结构直接存储 PLC 梯形图，省去了 PLC 向 AOV 图的转换过程，且使 PLC 绘制过程更加便捷、规范。文中提出的 AOV 图坐标更新算法简化了 AOV 图的各种编辑情况，使其只需考虑 AOV 图的结构变化，而无需对节点坐标的变化进行琐碎的处理。最后文中对 AOV 图生成二叉树的算法进行了修改，使其可适用于各种 AOV 图向二叉树的转换，并给出了具体的转换实例。



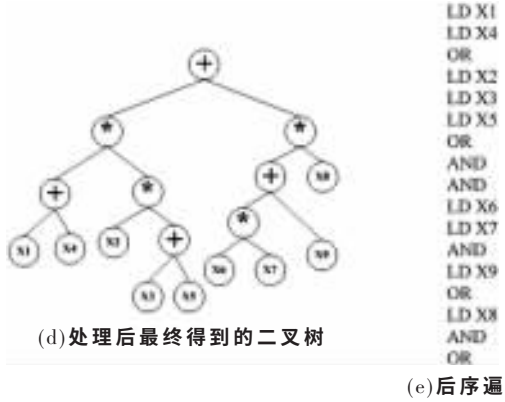
(a)梯形图程序



(b)AOV 图



(c)初始生成的二叉树




(d)处理后最终得到的二叉树

(e)后序遍历得到的指令表

图 5 转换实例

参考文献

[1] 俞锋达.PLC 编程软件的设计与下位机的仿真与实现[D].南京：东南大学，2008.

[2] 保慧.PLC 图形化编程系统的研究与实现[D].南京：东南大学，2008.

[3] 葛芬.水电自动化监控系统中 PLC 编程工具软件的设计与实现[D].南京：南京航空航天大学，2006.

[4] 崔小乐，周卓岑.可编程控制器的梯形图语言与语句表语言的互换算法[J].微电子学与计算机，2000，16(1)：26-30.

[5] 谭锦洁，程良鸿.嵌入式 PLC 中梯形图到 AOV 图的映射[J].计算机测量与控制，2004，12(10)：993-995.

[6] 傅亮，胡飞虎，刘乐，等.基于串并联归并的 PLC 梯形图向指令表转换算法[J].计算机工程与应用，2009，45(27)：72-118.

[7] 葛芬，吴宁.基于 AOV 图及二叉树的梯形图与指令表互换算法[J].南京航空航天大学学报，2006，38(6)：754-758.

(收稿日期：2012-03-22)

作者简介：

张惠杰，男，1987年生，硕士研究生，主要研究方向：嵌入式系统。

林伟敏，男，1988年生，硕士研究生，主要研究方向：嵌入式系统。