第 33 卷 第 13 期 计 算 机 工 程 2007 年 7 月
Vol.33 No.13 Computer Engineering July 2007
·软件技术与数据库· 文章编号：1000—3428(2007)13—0075—03 文献标识码：A 中图分类号：TP301.6

PLC 梯形图的广义表转换

林懋恺，王晓芳，林 亨
(清华大学工业工程系，北京 100084)

摘 要：提出了利用串并联归并算法以实现PLC梯形图到指令表的转换方法。该算法将梯形图转化为有向无环图，对图中的串并联关系进行分类归并，将串并联结构按层次存储在广义表中，根据广义表生成指令表。该算法克服了传统拓扑排序算法在梯形图结构复杂时产生误判的缺陷，增加了检查逻辑错误的功能。在最佳情况下，该算法的时间复杂度为O(n)，最差情况下为$O(n^2)$，与拓扑排序算法基本一致，有时略优于拓扑排序算法。

关键词：可编程控制器；梯形图；指令表

Transformation from PLC Ladder Diagram to Lists

LIN Maokai, WANG Xiaofang, LIN Heng
(Department of Industrial Engineering, Tsinghua University, Beijing 100084)

【Abstract】 To realize the transformation from ladder diagram to instruction list, a new algorithm is introduced. It translates the ladder diagram into directed acyclic graph, sorts and unites the serial and parallel relationship of the graph, and stores the serial-parallel structure in lists, generates the instruction list based on the lists. Traditional topological sort algorithm applied in the transformation does not always provide the correct instruction list when the structure of the ladder diagram is complex, while the new algorithm overcomes the flaw. The new algorithm provides the function of checking the logical faults in the ladder diagram. The time complexity of the new algorithm is O(n) in the best situation, $O(n^2)$ in the worst situation, which is basically the same with the topological sort algorithm. In normal condition, it is a little better than the topological sort algorithm.

【Key words】 programmable logic controller(PLC); ladder diagram; instruction lists

可编程控制器(programmable logic controller, PLC)在工业领域有着广泛的应用。PLC凭借着强大的定时、计数、逻辑运算、算术运算等能力在生产线自动化、柔性制造、大型分散系统控制等领域扮演着重要的角色[1]。

在PLC编程语言标准IEC61131-3 中，定义了 5 种PLC编程语言的表达方式[2]。其中，指令表和梯形图在PLC的设计和学习中应用最广。文献[4,5]阐述了基于AOV图(active on vertex)的拓扑排序算法。虽然拓扑排序能够解决部分梯形图结构提取，但是检查梯形图虚顶点和多路并联的方法并不理想，在复杂的电路判断上存在误判的情况。本文利用梯形图的性质提出一种算法，绕过虚顶点的判断，将梯形图结构转换为广义表，清晰地表示了梯形图层次，较好地解决了梯形图结构提取问题。

1 梯形图向语句表转换的拓扑排序算法

1.1 拓扑排序算法简述

梯形图向语言表转化的拓扑排序算法在 AOV 图中挑选入度为 0 的顶点，从图中移除并生成相应的语句，加入指令表。

PLC 梯形图与 AOV 图的对应关系如图 1 所示。首先寻找入度为 0 的结点，即结点 1，从图 1 中删去，同时删去弧<1，2>，将[LD 1]加入指令表中。此时入度为 0 的结点为 2，从图中删去，同时删去弧<2，3>、<2，4>、<2，5>，加入指令[AND 2]。此时结点 3~结点 5 入度为 0，任取一个结点，删去此结点和以此结点为弧尾的所有弧，将相应指令写入指令表。如此循环直至全部顶点都已删除，或者不存在入度为 0 的顶点为止。后一种情况说明有向图中存在环。

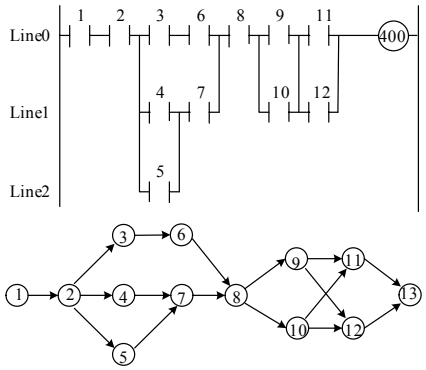

图 1 梯形图与 AOV 图

1.2 拓扑排序的优点和不足

拓扑排序算法比较直观、编程方便，在提取出 AOV 图后有良好的算法效率，时间复杂度为 O(n+e)。其中，n 表示结点个数，e 表示边数。然而，在 PLC 梯形图的转化过程中使用拓扑排序存在着一些缺陷，主要有 4 个方面：

(1)指令转化不稳定。从 1.1 节可以看到，在删去结点 2 之后存在 3 个入度为 0 的结点，这 3 个结点的处理顺序对指令表正确与否有很大关系。

(2)存在串并联关系的误判。例如图 1 中，结点 2 被移除后，结点 3~结点 5 的拓扑关系无法判断。3 个入度为 0 的结

作者简介：林懋恺(1985－)，男，学士，主研方向：PLC 仿真系统；王晓芳，讲师；林 亨，教授

收稿日期：2006-08-25 **E-mail：**lmk03@mails.tsinghua.edu.cn

点有可能是并联关系，也可能如图 1 所示没有直接的并联关系，而是间接并联。拓扑排序算法很难判断这样较为复杂的拓扑关系，因为这涉及到后继结点的检查。如果要检查后继结点，算法复杂度将大大提高、效率明显下降。

(3)查找虚结点困难。文献[4,5]都提出了虚结点的概念。例如在图 1 的 AOV 图中，结点 9、结点 10 的并联块和结点 11、结点 12 的并联块间应插入一个辅助的虚结点方便判断。但文献[1]没有给出虚结点的具体提取方法，文献[5]只给出了特殊形状的梯形图虚结点添加方法。由于虚结点对梯形图转化的正确性至关重要，因此虚结点提取的困难将直接影响到转化过程。

(4)无法查找被短路的结点。拓扑排序算法在遇到图 2 所示的情况将产生误判。图中的结点 B 实际上被短路，电流只经过<A，C>弧流向 C，B 结点为无用结点。在此情况下算法应该给出警告，同时删除结点 B，不生成相应代码。但拓扑排序会将 B 结点当作一个并联结点写入指令表。

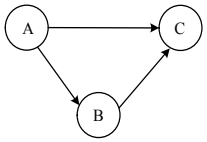

图 2　结点 B 被短路

用拓扑排序算法处理梯形图和指令表之间的转换，存在一些缺陷。另外，在处理过程中没有保留梯形图的层次关系。拓扑算法并不是最适合梯形图信息提取的算法，本文设计了一种串并联电路归并的算法。这种算法避开了虚结点的查找，在处理过程中生成 1 个与梯形图结构对应的广义表以表示电路的串并联关系。

2　串并联归并算法

2.1　AOV 图的数据结构表示

AOV图的数据结构主要包含邻接矩阵、邻接表、十字链表等[3]。文献[4,5,6]的AOV图数据结构，采用十字链表的存储方式。

在十字链表中，对应有向图中的每个顶点和每条弧都有一个结点，基本结构如图 3 所示。

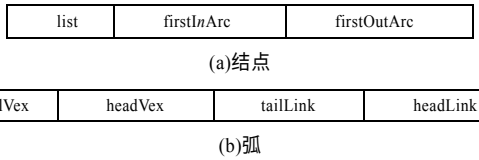

图 3 基本结构

2.2　梯形图向 AOV 图转换

(1)为每个 PLC 元件建立一个对应结点，在结点的广义表中存放元件的相应信息。以线性表 list 存储这些结点，同时初始化头结点 head。

(2)对每一个结点以深度优先法(depth first search, DFS)沿着梯形图连接线向右、向上、向下进行搜索，直到找到此元件的所有直接后继结点或搜索到右侧终止线。对结点 i 的每一个直接后继结点 jn，创建弧<i,jn>。弧插入时保证弧头结点编号最小的弧在后继弧链表首位。

(3)搜索所有直接连接到左侧起始线的结点，把这些结点加入到头结点的弧链表中。

在这样的操作中不需要判断和增加虚结点，只需要寻找所有结点的直接后继即可。

2.3　AOV 图的广义表转换

2.3.1　广义表的定义

广义表是线性表的一种推广。广义表一般记为

$LS=(\alpha_1,\alpha_2,...,\alpha_n)$

其中，n为广义表长度。与线性表不同，在广义表定义中，α_i可以是单个元素，也可以是广义表，分别称为广义表的原子和子表。

2.3.2　AOV 图的广义表转化算法

(1)串联扫描。在存储结点的线性表 list 中搜索所有满足以下 2 个条件的结点：

1)入度为 1。

2)直接前驱结点出度为 1。

记此结点为 Vj，其直接前驱结点为 Vi，则 Vi 与 Vj 之间为一一映射关系，反映在梯形图上为串联关系。此时归并 Vi 和 Vj：

1)用 Vj 的后继弧指针替代 Vi 的后继弧指针。

2)从 list 表中移除结点 Vj。

3)移除弧<Vi，Vj>。

完成结点归并后在 Vi 中记录广义表信息，分 2 类处理：

1)Vi 已储存一个最顶层为串联关系的广义表

①若 Vj 也是一个最顶层为串联关系的广义表，则将 Vi 子表的最末结点的指针指向 Vj 子表的首个元素，同时用 Vj 子表的尾指针替换 Vi 子表的尾指针。

②其他情况，只需简单地将 Vj 结点的广义表链到串联关系层的最后。

2)其他情况，具体包含以下 2 种：

①Vi 中的广义表 $Vi.ls$ 为初始的广义表原子。

②Vi 已存储一个最顶层为并联关系的广义表。

这时新建广义表结点 newLs，其类型为子表结点。然后将 Vi 和 Vj 链接到 newLs，将 newLs 顶层拓扑关系设置为串联关系。

串联扫描完成后图形中不存在直接串联关系的结点对。

(2)并联扫描。搜索所有出度大于 1 的结点，操作如下：

1)并联关系检查

记当前出度大于 1 的结点为V_0，其出度为$n(n>1)$，其直接后继结点分别为$V_1,V_2,...,V_n$。设V_1～V_n的直接后继集合为$S_1,S_2,...,S_n$。将V_1～V_n归入k个集合$SV_1,SV_2,...,SV_k$，使相同集合内所有结点的直接后继集相等。即任意集合$SV_i(i=1,2,...,k)$满足：

若结点Vi_a，$Vi_b\in SV_i$，则后继集$Si_a=Si_b$

简单地说，由于在相同集合SV_i内的结点在梯形图中有相同的直接前驱和相同的直接后继，因此这些结点在梯形图上是并联关系。

2)并联结点归并

以k个分类集合为单位进行归并，归并的过程同样分为结点合并和建立新广义表两个步骤。记集合SVi内的元素分别为$Vi_1,Vi_2,...,Vi_n$。

①从线性表list中移除结点$Vi_2,Vi_3,...,Vi_n$及它们所有的前驱弧和后继弧。

②这时新建广义表结点newLs，其类型为子表结点。然后将$Vi_1,Vi_2,...,Vi_n$链接到newLs，将newLs顶层拓扑关系设置为并联关系。

并联扫描完成后图形中不存在并联关系的结点集。

3)逻辑错误查找

令集合A=$\{V_1,V_2,...,V_n\}$，B=$S_1\cup S_2\cup...\cup S_n$。取C＝A∩B。若C≠Φ，则∀ $Vi\in$C，有：

①$V_i\in$A，即V_i是V_0的一个直接后继。

②$V_i\in$B，设$V_i\in S_j$，则V_i是V_j的一个直接后继。

V_0对应图 2 中的A结点，V_j对应B结点，V_i对应C结点。显然，V_j为无效结点。此时记录警告信息并从线性表lists中移除

结点V_j及其所有前驱弧和后继弧。

重复(1)、(2)，直到所有结点归并成为一个结点(全部归并到头结点)。图 1 梯形图的归并过程如图 4 所示。归并结束后，头结点中的广义表为 PLC 梯形图对应的广义表。

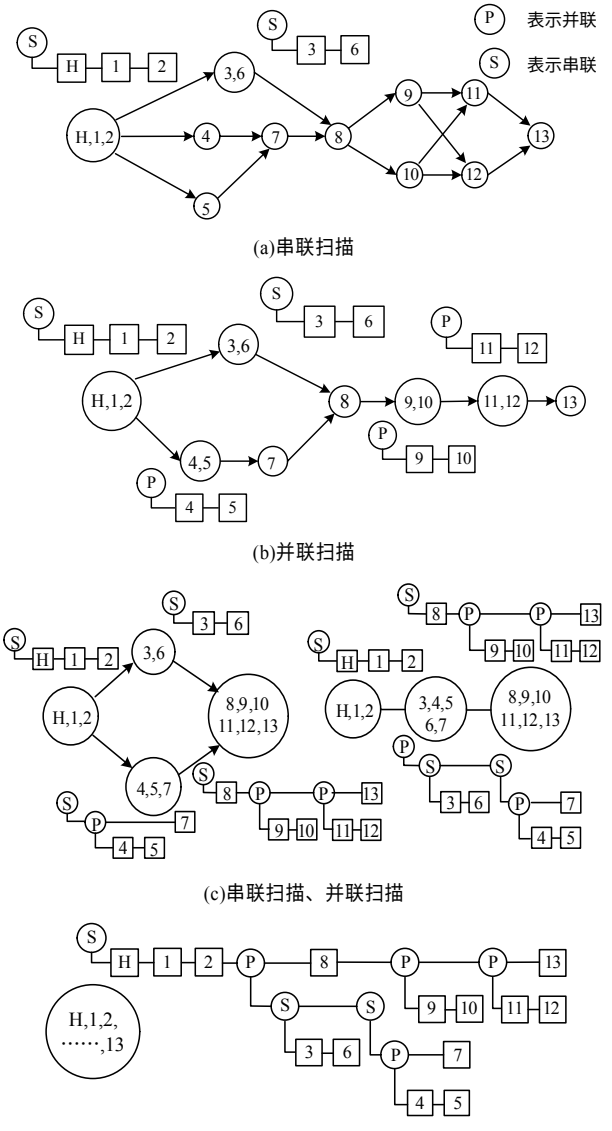

图 4　串并联归并过程

提取图 4 所示的广义表之后，串并联关系已经非常清晰。图中以圆形表示的子表结点 *S* 和 *P* 分别代表着一个块与和一个块或，以方形表示的原子结点代表着单个 PLC 元件。

提取时可以从广义表的头结点开始扫描，遇到表示串联或并联的子表则开始一个块与/块或，然后一一检查此子表的元素，若为原子结点，则产生 AND 或 OR 语句加入指令表；若为子表，则递归产生子表的子指令表，在子指令表开头加入块与/块或指令后加入总指令表中。如此循环直到产生完整的梯形图指令表。

3　算法复杂度分析和比较

3.1　拓扑排序算法的时间复杂度

经典的拓扑排序算法对有 *n* 个顶点和 *e* 条弧的有向图而言，求各顶点入度的时间复杂度为 O(*e*)；建立活动顶点栈的时间复杂度为 O(*n*)，总的时间复杂度为 O(*n*+*e*)。

3.2　串并联归并算法的时间复杂度

设 *i* 为串联关系的数量，即广义表中所有串联关系层的结点数目。例如在图 4 中，*i*=12。每一对串联结点需要进行一次串联归并，时间复杂度为 O(*i*)。

设 *j* 为并联关系的数量，即广义表中所有并联关系层的结点数目。若结点 *V* 有多个后继结点，则这些后继与 *V* 之间必然有一个多路接点，如图 5 所示。若一个结点 *V'* 与 *V* 的一个后继有连接，则 *V'* 一定连到该多路接点，必然连接到 *V* 的所有后继，且有如下普遍结论：在梯形图中 *Vi* 和 *Vj* 的后继结点都相同，当且仅当 *Vi* 与 *Vj* 有一个相同的后继结点。

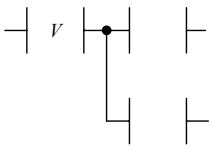

图 5　多路连接

在*j*个并联关系的归并过程中，每个待归类结点只需与已产生的类比较 1 次。最好情况下所有结点归为同一类，需比较*j*次，时间复杂度为O(*j*)。在最坏情况下所有结点各成一类，比较(*j*－1)*j*/2 次，时间复杂度为O(*j*^{2})。

在逻辑查错方面，假设顶点V_i的后继为V_j，$j=1,2,\ldots,k$。则A=$\{V_1,V_2,\ldots,V_n\}$，B=$S_1\cup S_2\cup\ldots\cup S_k$。设B中元素数量为*m*，则求交集C＝A∩B时间复杂度为O(*n*+*m*)。由a≤*n*、b≤*n*可知，求交集的时间复杂度为O(*n*)。

广义表结点的连接和指令表的提取分别需要进行 *i*+*j* 次操作，时间复杂度 O(*i*+*j*)。

算法各部分的时间复杂度整理见表 1。

表 1　时间复杂度整理

项目	串联归并	并联归并	逻辑查错	广义表操作
时间复杂度	O(*i*)	最好: O(*j*) 最差: O(*j*^{2})	O(*n*)	O(*i*+*j*)

总时间复杂度最好情况下为

O(2*i*+2*j*+*n*)

其中，*i*＋*j*＝*n*+δ，δ为串并联关系总数，即广义表中的子表总数。显然，δ≤*n*。最佳情况总时间复杂度为O(*n*)。最差情况下为O(2*i*+*j*+ *j*^{2}+*n*)。由于*j*与*n*数量级相同，因此最差情况总时间复杂度为O(*n*^{2})。

3.3　算法复杂度比较

拓扑排序的算法复杂度为O(*n*＋*e*)。当边稀疏时，算法复杂度趋近于O(*n*)，当边稠密时，算法复杂度趋近于O(*n*^{2})。

本文提出的串并联归并算法复杂度取决于梯形图串并联层次的混合程度。若电路中串并联混合程度很高，例如在并联的支路上有串联，而串联的各个部分又有并联，如此循环，则时间复杂度趋近O(*n*^{2})；若串并联关系清晰、层次较浅，复杂度趋于O(*n*)。串并联层次的混合程度深的情况正是在拓扑排序算法中容易出错的地方，串并联归并算法为了得到正确的判断而付出较大时间复杂度的代价。梯形图串并联关系层次一般较浅，在这样的情况下串并联归并算法的效率高于拓扑排序算法。串并联归并算法在未增加时间复杂度的基础上克服了拓扑排序算法中的误判问题，增加了逻辑查错功能，同时在一般情况下算法的效率优于拓扑排序算法。可以说，算法效率、正确性、可靠性、健壮性同时得到了保证。

（下转第 95 页）