

# 程序流程草图的存储表示及自动翻译算法<sup>①</sup>

## Representation and Storage of Sketching Flow Diagram and Its Automatic Translate Algorithm

何 骅<sup>1,2</sup> 诸 彬<sup>3</sup> 徐从富<sup>1</sup>

(1.浙江大学 计算机科学与技术学院 浙江 杭州 310027; 2.浙江教育学院 现代教育技术系

浙江 杭州 310012; 3.杭州师范大学 信息科学与工程学院 浙江 杭州 310036)

**摘要:** 将手绘草图识别技术融于在线手绘程序流程草图的识别中,设计了一个存储图元的节点结构,提出基于上下文的在线手绘程序流程草图自动翻译算法,实现在线手绘程序流程草图的存储、识别和到 C 语言代码的自动翻译和运行,实验证明该方法具有较高的识别效率。

**关键词:** 草图 程序流程图 逻辑判别 上下文 栈

### 1 引言

手绘草图是人类一种自然而直接的思路外化和交流方式<sup>[1]</sup>。手绘流程草图作为手绘草图的一个分支,因其具有直观、逻辑严密、思路清晰等优点,被广泛应用于日常生活。尤其是在日常教学中,教师常常使用流程图来描述问题、讲解知识。在计算机专业的教学当中,很多课程均涉及到流程图的绘制。特别是高级语言程序设计课程,常使用程序流程图来说明算法过程。若用草图来绘制程序流程图,并且实现流程图的自动识别和代码生成,不仅符合人的思维习惯,也将大大方便程序设计的教学,同时也可应用于程序员的快速程序设计和编码。

目前,国内<sup>[2-5]</sup>国外<sup>[6-8]</sup>对手绘草图识别的研究已有较大进展,可较好地识别程序流程图中的简单图元符号<sup>[9]</sup>。但是对程序流程图控制结构的判别,以及代码的自动翻译的研究较少。本文设计了一个存储图元的节点结构,在图元识别的基础上,提出基于上下文的程序流程草图自动翻译算法,实现了在线手绘程序流程草图到 C 语言代码的自动翻译。

### 2 程序流程图图元

程序流程图是历史最悠久、使用最广泛的描述算法的工具,是算法的图形表现形式。它使用几何图形、

流程线和文字说明来论述一个算法。因其直观、易懂、便于初学者掌握使用的特点而被广泛使用。

程序流程图使用特殊的符号来绘制,主要包括如表 1 所示的几个符号(以下简称图元)。

为了便于提高识别效率和配合用户的习惯,本文简化了圆角矩形,这样用户在绘制程序流程图时可省略起始框和终止框,如有特别需要可使用圆来替代,本文系统可自动识别开始或结束模块。该简化操作不仅符合多数用户的习惯,也符合教学设计中快捷简便的原则。

图元节点结构。为了方便流程图图元的存储、遍历和控制结构的转换,本文设计了如图 1 所示的节点结构。在草图图元识别的基础上,根据程序流程图中构成不同控制结构的上下文信息,对相关的具体数据字段进行赋值,分别利用栈 S1 和 S2 进行程序流程图的存储和遍历,从而实现程序控制结构的识别。

### 3 流程图中控制结构的表示和存储

Bohm 和 Jacopini<sup>[10]</sup>的研究表明,所有程序只要使用三种控制结构就可以编写出来,分别是顺序结构、选择结构和循环结构。

手绘程序流程草图中各个图元的上下文信息构成

① 基金项目:国家自然科学基金项目(60402010);国家 863 计划专题项目(2007AA01Z197)

收稿时间:2008-09-30

了不同的控制结构。

表 1 程序流程图常用符号







	圆角矩形	起止框
	平行四边形	输入输出框
	菱形	判断框
	矩形	处理框
	线	流程线
	圆	出入口点(流程图较大时使用)



图 1 节点结构

### 3.1 顺序结构的表示和存储

顺序结构是手绘程序流程图的基础部分，由基本图元矩形、平行四边形和圆构成。两个基本图元之间用 OUT1 来表示向下连接，如图 2 所示。

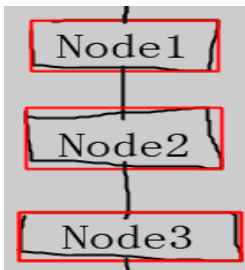


图 2 顺序结构

其结构化描述如下：

Node1.out1 = Node2;

Node2.out1 = Node3;

### 3.2 选择结构和循环结构的表示和存储

菱形和矩形、平行四边形的组合则可根据图元出入度的不同构成分支结构或循环结构。如图 3、4 所示，当菱形模块入度为 1，出度为 2，且出度的两个方向最终相交时表示 IF 选择结构；当模块入度为 2，出度为 2，且出度方向有一条边最终指回该模块时表示 WHILE 循环结构。利用这一特征，我们可以分别对顺序、选择和循环结构进行表示和存储。具体方法如下：

(1) 当识别草图图元为菱形(即判断框)时，压入栈 S1

(2) 若栈 S1 的栈顶元素的 out4 和 out5 指向节点为线段连接的两端图元时，进入存储 If 模块流程，如图 3 所示。创建完毕后需进行出栈更新操作。

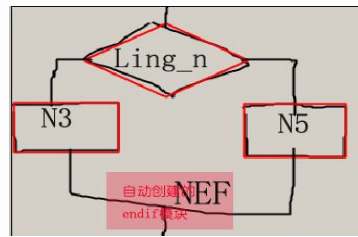


图 3 分支结构

其结构化描述如下：

Ling\_n.out1 = N3;

Ling\_n.out4 = N3; (若 N3 下继续有节点添加则,OUT4 自动调整到下一个节点)

Ling\_n.out2=N5;

Ling\_n.out5 = N5; (同 out4)

Ling\_n.out3=NEF; ( NEF 为判断为 IF 模块时创建的结束节点 )

(3) 若终点指向 S1 栈顶元素，且 S1 栈顶元素有出度时，进入存储 WHILE 模块流程，如图 4 所示。创建完毕后需进行出栈更新操作。

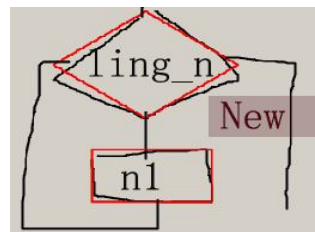


图 4 循环结构

其结构化描述如下:

Ling\_n.name="WHILE",

Ling\_n.out1 = n1;

Ling\_n.out2 = New;(New 为判断为 WHILE 模块时创建的结束节点)

## 4 程序流程图自动翻译算法

### 4.1 节点更新算法

流程图程序逻辑的自动识别主要基于上下文的组合信息并结合流程图领域知识。本文创建了一个通用的节点,把每个基本图元作为基本节点存储保存,构造一般的流程图逻辑关系图并设置相关的出入方向,使之能够正常地存储和遍历访问。流程图节点的更新操作是在栈 S1 非空时创建图元所需的操作,具体步骤如下:

步骤 1. 判断是否要对栈 S1 的当前栈顶节点的 OUT4 和 OUT5 字段进行更新。如果当前创建的节点有前继节点,可通过与之相连接的线段节点的指向得知它的前一节点。

步骤 2. 判断该节点的前一个节点是否和栈顶的 OUT4 或 OUT5 相同。若相同,则把 OUT4 或 OUT5 更新到当前节点。

步骤 3. 创建 IF 模块或 WHILE 模块时的出栈更新操作。判断当前出栈的节点,若出栈后栈非空,则查看已出栈节点是否是当前栈顶节点的 OUT4 或 OUT5 指向,并将此节点进行数据结构更新到出栈节点的结束节点(即 IF 模块的 OUT3 指向节点,WHILE 模块的 OUT2 指向节点)。

### 4.2 逻辑识别算法

通过对流程图的构造,可确定有以下三种节点模块类型:顺序(向下执行)模块、IF 模块、WHILE 模块。采用递归访问法,模块的结束条件是当前节点的下一个节点为空(即已访问到最后的节点)输出结束代码,具体执行步骤为:

步骤 1. 从初始节点出发,对存贮结构进行遍历。

步骤 2. 如果是顺序节点,则加入代码后访问其下一个,并标记为已访问。

步骤 3. 若访问到 IF 模块节点,先将该模块中的判断框压入栈 S2,递归访问 OUT1 指向节点,当访问到 IF.OUT3 指向的节点时,结束 OUT1 方向的访问并返回。并将 IF 模块的 OUT1 方向标记为已访问。接着访

问 IF 模块的 OUT2 指向节点,当访问到 IF.OUT3 模块时,结束 OUT2 方向的访问,将 IF 模块出栈,并输出 IF 结束标记。继续访问 IF 结束节点的下一节点。

步骤 4. 若访问到 WHILE 模块节点,先将该模块中的判断框压入栈 S2,标记 WHILE 模块的访问计数为 1,递归访问 OUT1 指向节点,当访问到的节点指向栈顶时结束 OUT1 方向的访问,将 WHILE 模块出栈,并输出 WHILE 结束标记。继续访问 WHILE 结束节点的下一节点。

## 5 算法验证和实验结果

实验平台 Windows XP, 实验环境 Microsoft Visual Studio 2005, 用户输入设备为 Tablet PC。利用 Tablet PC SDK 快速采样得到组成草图的每个 stroke 信息组。在实现基于草图的流程图基本图元识别的基础上,根据以上算法采用递归遍历存储流程图数据结构的方法,使程序流程草图自动转化为简单 C 语言代码。实验显示,基本逻辑转换成功,总体识别效果较好。

实例: 求 100~200 间的能被 3 整除的数。

首先对这个问题进行分析,程序设计的算法流程图绘制如图 5(a)。流程图创建好之后,对该流程图的数据结构进行遍历操作,得到如图 5(b)的 C 语言代码。

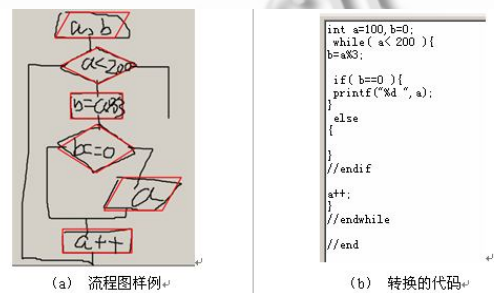


图 5 实例流程图及其代码自动翻译

代码自动翻译后,还可调用外部编译器输出结果,如图 6 所示。

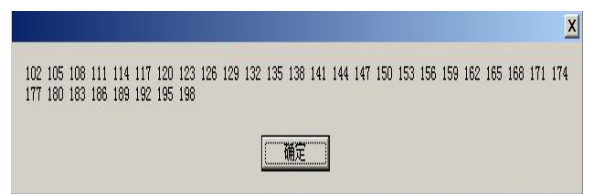


图 6 外部编译器结果

## 6 结束语

根据在线手绘程序流程草图,设计了一种用于存储图元的节点结构,并以此为基础进行流程图图元的存储、在线识别和更新。并根据上下文信息,实现了逻辑控制结构的识别。可将程序流程草图自动翻译成对应的C语言代码,并编译运行输出结果。

本文系统目前所能判别的基本图元比较单一,在判断程序流程图控制结构时,采用栈技术,若用户采用非习惯性思维画图,可能会导致栈顶元素无法正确判别,在一定程度上影响识别结果。

### 参考文献

- 1 Fish J, Scrivener S. Amphifying the mind's eye: sketching and visual cognition. Leonardo, 1990,23(1): 117-126.
- 2 栗阳,关志伟,戴国忠.笔式用户界面开发工具研究.软件学报, 2003,14(3):392-400.
- 3 孙正兴,冯桂焕,等.基于草图的人机交互技术研究进展.计算机辅助设计与图形学学报, 2005,17(9):1889-1899.
- 4 蒋维,张斌,孙正兴.基于自适应 HMM 的在线草图识别方法.计算机科学, 2005,32(5):185-189.
- 5 谢强,冯桂焕,孙正兴.基于上下文的在线草图识别方法.计算机科学, 2007,34(3):216-219.
- 6 Kara L B, Stahovich T F. Hierarchical parsing and recognition of hand-sketched diagrams. Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology, 2004.
- 7 Kara L, Stahovich T. Sim-U-Sketch: A Sketch-Based Interface for Simulink. Proceedings of Advanced Visual Interfaces. 2004:354-357.
- 8 Shilman M, Viola P. Spatial recognition and grouping text and graphics. EURO GRAPHICS Workshop on Sketch-based Interface and Modeling, 2004.
- 9 张小亮,孙根正,廖达雄,王淑侠.基于几何特征的在线手绘流程图识别.计算机辅助工程, 2007,3:29-33.
- 10 Bohm CG, Jacopini. Flow Diagrams, Turing machines, and languages with only two formation rules. Communications of the ACM, 1966,9(5): 336-371.