基于语法树和JavaCC的程序题自动评分系统

屠方博,杨志强
(同济大学 电子与信息学院,上海 201804)

摘 要:为了提高程序题自动评分的准确性,及解决传统评分方法无法从语法结构和语义角度衡量错误的学生程序与正确答案之间的相似度,提出了一种基于抽象语法树匹配的程序题自动评分方法。文中以JavaCC技术为核心,首先通过词法分析、语法分析和语义分析生成错误列表和抽象语法树的中间表示,然后通过语法树切片匹配得分,最后和错误列表结合给出评分结果。文中详细论述了各个模块的设计方法,着重讨论了抽象语法树生成并匹配的细节,设计并实现了一个传统方法与语义分析结合的C++程序题自动评分系统。通过对实际考试的结果进行实验,进而验证了该系统的实用性与有效性。

关键词:抽象语法树;程序评分;JavaCC
中图分类号:TP39 **文献标识码**:A **文章编号**:1673-629X(2012)01-0126-03

Grading of Programs Using Abstract Syntax Tree & JavaCC

TU Fang-bo, YANG Zhi-qiang
(School of Electronics and Information, Tongji University, Shanghai 201804, China)

Abstract: A system for grading student programs is designed to enhance the accuracy, and solve the problem that traditional automatic grading approaches can not evaluate the similarity of the source code to correct solutions by using syntax structure and semantic analysis. Present a tool for quickly comparing the source code of student programs and correct solutions. The approach is based on partial abstract syntax tree matching (AST), and using JavaCC to generate the AST, then making the score by Tree Cuts and combine the parse error. In this paper, designed and implemented a grading system on C++ programs, and the semantic analysis is used in conjunction with traditional approaches. And then, verify the practicability and effectiveness of the system.

Key words: syntax tree matching; grading of programs; JavaCC

0 引 言

随着计算机等级考试的普及,程序设计的评分越来越要求迅速而准确,但对程序题的评分是计算机基础教育的一个重点和难点,所以设计并实现一个程序题的自动评分系统来取代手工对程序评分,可以确保成绩公平合理并给教师快速提供信息反馈。

程序设计语言语法的复杂性及实现方法的多样性,造成了程序评分系统语义理解上的缺陷。目前,很多程序作业的评分系统和程序竞赛评分系统投入使用,也有很多人在研究评分方法,如Nott的CourseMarker[1]系统,北航的CourseGrading[2]系统,哈工大[3]基于SDG的评分方法,传统的评分系统按照方法可以分为两类:软件质量度量和动态测试。但都存在无法分析程序语义的缺陷。

收稿日期:2011-05-30;**修回日期**:2011-09-04
作者简介:屠方博(1986-),男,浙江乐清人,硕士研究生,研究方向为模式识别与智能系统;杨志强,教授,研究方向为模式识别与图像、语音信号处理。

文中的目的是设计和实现一个提供大量学生进行程序设计语言考试自动评分的系统,针对目前已存在的问题,提出了基于抽象语法树和JavaCC的程序评分方法,与传统动态测试方法结合,来实现对程序评分准确性的提高。该方法是基于观察C++程序类名与函数名称的相关性和匹配语法树的结构。在这个过程中,计算一个类型名和变量名的双映射来决定这个结构的相似度。文中首先介绍了整个系统的模型架构。然后详细解释利用JavaCC[4]工具进行语法语义错误分析,及生成抽象语法树切片的方法和计算评分的算法。最后,对已有的学生程序进行评分实验,分析实验结果。

1 系统概述

评分系统主要由三个模块组成,分别定义为语义分析模块、语法树切片匹配模块和界面生成模块。其中,语义分析模块主要进行语义解析生成语法树并统计错误,界面生成模块采用Swing生成界面,图1提供了评分系统的主要方法的抽象描述。工具通过分析测

试程序代码的语法书切片与正确程序的相似度,以及代码中所含有的语法语义错误综合给出评分结果。首先对测试程序和正确程序进行语法语义分析来产生抽象语法树(ASTs),然后通过抽象语法树的匹配[5]来生成测试程序与模板程序的语法树切片,最后由语法树切片进行计算得出相似度分析的评分结果。

在这一部分描述抽象语法树的匹配算法,如何进行相似度计算,并介绍了实现过程。

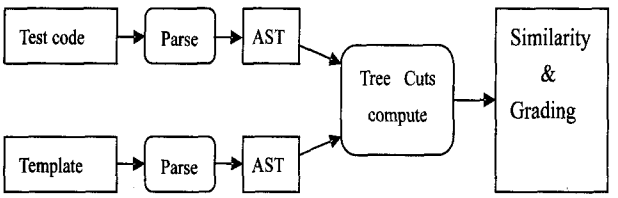

图 1　抽象语法树匹配概要图

1.1　语法树切片

编程题的自动评分是评分系统的一个热点和难点,程序语义分析的难度较大,一直没有很好的解决方法。文中采用基于语法树的抽象表示来进行语义分析。语法树是程序结构的中间表示,它具有比源程序更加抽象的语义表示,通过语法树可对代码进行静态测试[6]及软件度量[7],从而较好地对代码进行分析。

在文中,生成代码的中间表示抽象语法树后,专注于对语法树的特定部分进行匹配,这里称其为语法树切片(Tree Cuts)。一个语法树切片(TC)[8]是以原抽象语法树的任意节点为根,但无需延伸到叶子节点的子树。在语法树匹配的算法中,这些语法树切片就是测试程序和模板程序各自抽象语法树的公共部分,由于叶子节点为终结符或不含分数的非终结符,可略去不作为评分依据。定义一个 AST 子树集是语法树切片集的一个子集。

1.2　语法树切片生成算法

语法树切片的匹配符合以下定义:

定义　语法树切片 $TC_1 \in AST_1$, $TC_2 \in AST_2$, v_1, v_2 分别是 TC_1 和 TC_2 的任意节点,TC_1 和 TC_2 相等当且仅当:

1) v_1, v_2 无兄弟节点且类型相同;

2) v_1, v_2 有兄弟节点则都有父节点且父节点类型相同;

3) v_1 和 v_2 是一一对应的关系。

该抽象语法树切片生成算法基于 3 个主要步骤:

1)构建抽象语法树(ASTs);

2)根据函数、类作用域对语法树进行排序;

3)搜索语法树匹配并生成切片。

图 2 给出了语法树匹配算法的伪码。

1.3　设计实现

评分系统使用 Java Swing 架构,在词法、语法分析

```
Procedure GenerateCuts(Pro1, Pro2)
A1 ← set of all ASTs in Pro1
A2 ← set of all ASTs in Pro2
Global TreeCuts ← ∮
For each ASTs (a1, a2) ∈ (A1, A2)
Do MATCH_AST (a1, a2)

Procedure MATCH_AST (AST1, AST2)
For each (root1, root2) ∈(AST1, AST2)
do { If (root1 = root2)
       TreeCuts ← root2
       MATCH_AST (subtree(root1), subtree(root2))
     Else
       MATCH_AST (subtree(root1), root2)
```

图 2　生成语法树切片算法

和语法树生成中使用 JavaCC 工具,JavaCC 是一个生成 Java 代码的词法语法生成器,可以使用它来辅助构造词法语法生成器[9],文中,使用它的 JJtree 工具[10]来帮助设计语义结构,JJtree 适用于建立语法分析树和预处理,通过编写 JJt 文件,可以更好地改善 JavaCC 的逻辑结构。通过使用 JJtree 将测试程序转换为抽象语法树后,那么对程序代码的评分也就转换成了对语法树切片的相似度评分,从而,使用工厂和访问者模式来生成各语法树节点并遍历语法树来生成评分标准。

文中,采用 Swing 生成界面,实验以观察结果,可以作为以后生成 BS 系统的基础。本系统首先定义语法词法文件 CPPParser. jjt。然后再加入语法制导的翻译,通过以上这两步可以找出语法和语义错误,单独地加入错误模块,可以避免过多关联错误的生成,设置错误上限,对超过上限的错误数统一扣除相应的分数。

在 JJTree 文件中设置节点的生成,在 JJTree 的文件中,采用两种生成节点的有效方法:

1)条件节点,#ConditionalNode(BOOLEAN EXPRESSION),设置条件节点可以对符合模板程序得分点的语法树进行生成。

2)不生成节点,#void,设置后可以对无关评分的语法树节点在评分中不予生成,提高评分的准确率。

在实际操作中,基于以上步骤,并和动态测试的方法作了对比,实现了语法树切片生成的算法,并实现了整个自动程序评分系统。再通过实验来分析系统的可行性和实用性。

2　应用和实验

本方法可以简单地应用于各种程序语言的考试系统中,也可以作为判断代码抄袭和程序版本信息的一种补充。为了验证方法的可行性和系统的性能,将采

用本方法的系统应用于大学计算机基础教育中的 C/C++程序设计课程的程序题评分中。本测试在 4 场考试中都采用一个语法树模板进行评分，题目难度适中。分别采用动态测试评分，语法树评分，来对和已经手工评阅的分数进行对比。

实验分析：

在 2010-2011 学年第一学期期末同济大学举行的大学计算机基础课程考试中，采用本系统对 C/C++程序设计课程的程序题部分进行阅卷，实验结果如表 1 所示。

其中，“可评分份数”是指使用动态评分，语法树评分可以评分的源码数。“可评分率”计算公式如下：

$$\text{可评分率}=\frac{\text{可评分源码数}}{\text{源码总数}} \tag{1}$$

“人工评分份数”是指从待阅源码中随机抽取一定数量的程序题进行人工阅卷，“平均误差”为采用动态评分，语法树评分所得分数和采用人工评分所得分数之差的平均值，其计算公式如下：

$$\text{avgErr}=\frac{\sum_{i=1}^{n}(\text{auto}_i-\text{manul}_i)}{n} \tag{2}$$

其中，n 指人工评分份数，auto_i 是指对第 i 个源码采用自动评分程序阅卷所得分数，manul_i 是对第 i 个页面采用人工阅卷所得分数。

表 1 实验结果

场次	学生人数	源码数	可评分率	误差分析		
				手工份数	平均误差（AST）	平均误差（动态测试）
1	183	3	98.06%	20	1.00	2.35
2	217	2	99.72%	20	0.55	1.4
3	194	2	99.33%	20	0.8	2.05
4	65	2	100%	20	1.75	3.65

通过对以上数据的分析得出以下几点：

（1）平均可评分率达到 99.28%，个别源码无法通过自动阅卷程序评阅是由于：

①源码后缀错误。程序只对后缀名为. c 或. cpp 的源码文件进行评阅；

②源码文件被损。个别源码文件因未知原因被损坏，导致无法评阅。

（2）单个场次中的平均误差最大不超过 2，测试的 4 场考试中的平均误差仅为 1.02，这说明采用本系统对程序进行评分与采用人工评分所得结果的误差是令人满意的，语法树程序评分完全可以代替人工阅卷。在与动态测试[11,12]的比较中，由于动态测试对不能成功编译通过和测试数据全不匹配的程序只能给予零分，不能按所编写的程序结构分数点评分，所以有较大误差，也说明这种方式的评分效果比动态测试的评分效果好。

3 结束语

文中提出一种基于抽象语法树和 JavaCC 实现的评分方法，从词法、语法、语义制导翻译中检测出错误，由 JJTree 实现了语法树的生成，再由语法树切片中提取出程序分数的属性，实现了一个程序题自动评分系统。文中还研究了语义分析的方法来对程序的语义进行评分，能更好地将教师从复杂的手工评分方法中解放出来。在与动态测试方法的对比过程中，也显示出有更好的语义理解能力，能从语句中抽象出语义层次。

文中实现的方法可以作为语义评分的基础，据此提出的语义方法评分也较符合手工建立的评分标准，作为计算机基础教育的一部分也具有实际的应用意义。

参考文献：

[1] Higgins C A, Gray G, Symeonidis P, et al. Automated assessment and experiences of teaching programming [J]. ACM Journal on Educational Resources in Computing (JERIC), 2005, 5(3): 5-26.

[2] Zhao Changhai, Yan Haihua, Song Cong. A System for Automatic Assessment and Plagiarism Detection of Student Programs[C]//International Conference on E-Business and E-Government (ICEE). Guangzhou: [s. n.], 2010: 3619-3624.

[3] Wang Tiantian, Su Xiaohong, Wang Yuying, et al. Semantic similarity-based grading of student programs[J]. Information and Software Technology, 2007, 49(2): 99-107.

[4] Kodaganallur V. Incorporating Language Processing into Java Applications: A JavaCC Tutorial[J]. IEEE Software, 2004, 21(4): 70-77.

[5] Funaro M, Braga D, Campi A, et al. A Hybrid Approach (Syntactic and Textual) to Clone Detection[C]//ACM International Workshop on Software Clones (IWSC). [s. l.]: [s. n.], 2010: 79-80.

[6] 高传平，谈利群，宫云战. 基于抽象语法树的代码静态自动测试方法研究[J]. 北京化工大学学报：自然科学版，2007，34(z1)：25-29.

[7] 牛家浩，黄志球，张 静. 基于抽象语法树的软件度量工具的设计与实现[J]. 计算机应用，2003，23(10)：39-41.

[8] Neamtiu I, Foster J S, Hicks M, et al. Understanding source code evolution using abstract syntax tree matching[J]. ACM SIGSOFT Software Engineering Notes, 2005, 30(4): 1-5.

[9] 姚 砺，束永安. 用 JavaCC 构造编译器的方法[J]. 计算机工程，2003，29(9)：39-41.

[10] 周 健，孙丽艳. 用 JavaCC 和 JJTree 构造扩展模式文档解析器[J]. 计算机技术与发展，2008，18(9)：87-90.

[11] 张 量，詹国华. 开放式、智能化计算机考核阅卷系统的设计与实现[J]. 计算机工程与应用，2001，37(10)：108-110.

[12] 思 丹，袁春风. 主观试题的计算机自动批改技术研究[J]. 计算机应用研究，2004，21(2)：181-185.