****

本科毕业设计(论文)

GRADUATION DESIGN(THESIS)

|  |  |
| --- | --- |
| 题 目： | 基于约束逻辑编程的C程序流程图自动生成 |
| 学生姓名： | 张琦 |
| 指导教师： | 任胜兵 |
| 学 院： | 软件学院 |
| 专业班级： | 软件1406班 |

本科生院制

2018年6月

基于约束逻辑编程的C程序流程图自动生成

摘要

将源程序转换为控制流图是软件工程领域中逆向工程的研究内容之一。利用控制流分析技术对源程序进行控制流分析，并生成其所对应的控制流图，控制流图在程序的分析和理解领域有着广泛的应用。

本文以C程序源代码作为研究对象，并利用静态分析技术，来构造程序的控制流图。首先在语言文法的基础上，对C语言的各种语句结构和用法进行分析和研究，重点分析其各种语句结构的控制流走向及其特点，然后设计相应的从源程序自动生成其所对应的程序控制流图的算法；根据程序控制流图生成算法，在分析器自动生成工具ANTLR的基础上，自动的生成其所对应的控制流分析器，然后利用生成的控制流分析器对给定的C源程序进行分析便可得到其所对应的控制流图；此外为了把生成的控制流图以直观的形式展现给用户，我们还使用ECLiPSe约束逻辑编程自动生成程序控制流图，并用ECLiPSe提供的API进行可视化处理，此基础上最终实现一个面向C语言源代码的程序控制流图自动生成的实验系统。

**关键词：**控制流图 静态分析 ANTLR ECLiPSe

**Automatic generation of C program flowchart based on constraint logic programming**

**ABSTRACT**

Converting a source program into a control flow graph is one of the research contents of reverse engineering in the field of software engineering. The control flow analysis technology is used to analyze the control flow of the source program and generate its corresponding control flow graph. The control flow graph has a wide range of applications in the field of program analysis and understanding.

This paper takes the program source code as the research object, and uses the static analysis technology to construct the control flow graph of the program. Firstly, based on the language grammar, it analyzes and studies various sentence structures and usages of the Sui language. It focuses on analyzing the control flow and characteristics of various sentence structures, and then designing the corresponding source code to automatically generate the correspondences. The program controls the flow graph algorithm; according to the program control flow graph generation algorithm, based on the analyzer automatic generation tool ANTLR, it automatically generates its corresponding control flow analyzer, and then uses the generated control flow analyzer for the given The source code of the program can be analyzed to obtain its corresponding control flow graph. In addition, in order to present the generated control flow graph to the user in an intuitive manner, we also use the ECLIPSE constraint logic to programmatically generate a program flow chart and provide it with ECLIPSE. The API performs visual processing. Based on this, an experimental system for automatically generating program control flow graphs for C language source code is finally realized.

**Key words：**Control flow diagram Static analysis ANTLR ECLiPSe

1. 研究背景及意义

1.1 课题研究背景

随着软件规模的扩大，软件的可靠性显得越来越重要，对软件进行测试是保证软件质量的重要环节，由于在软件的开发后期发现错误进行修改是早期发现错误进行修改所需成本的很多倍，因此在动态测试前对软件进行静态测试可以尽早的发现软件中的缺陷，提高产品的质量，加快开发速度，减少软件的开发成本。静态分析主要包括代码检查、静态结构分析、代码质量度量等等，而软件维护、程序分析、 逆向工程和转换工具几乎都依靠静态源代码分析作为基础。因此，研究促进程序理解的软件工程方法和工具就显得非常必要，逆向工程正是在这种需要下产生的软件工程方法。逆向工程着眼于分析软件并用抽象形式将其表现出来，使其便于理解。 逆向工程可用于软件维护、再工程、重用以及文档化等许多用途。

对计算机程序行为进行自动分析的过程，主要包括静态分析和动态分析两种策略。程序静态分析是在不执行程序的情况下对其进行分析的技术，简称为静态分析。而程序动态分析则是另外一种程序分析策略，需要实际执行程序。大多数情况下，静态分析的输入都是源程序代码，只有极少数情况会使用目标代码。静态分析这一术语一般用来形容自动化工具的分析，而人工分析则往往叫做程序理解。

静态分析越来越多地被应用到程序优化、软件错误检测和系统理解领域。Coverity Inc.的软件质量检测产品就是利用静态分析技术进行错误检测的成功代表。国内某软件公司的闪蝶（BlueMropho）代码分析平台，是利用程序静态分析技术专注于大型机遗留系统的代码理解领域，尤其擅长分析千万行代码规模级的COBOL系统。

1.2 课题研究意义

构造程序的控制流图是软件工程领域的重要研究课题之一，此外控制流图在程序切片、软件测试中也有着广泛的应用。在实际的应用中，为了更好地利用控制流分析结果，如何将分析结果以更直观的方式显示出来也是目前控制流分析技术研究的重点内容之一，本项目也将对其进行深入的分析和研究。

本项目的研究内容具有重要的科学价值。对程序的控制流图进行相应的分析可以帮助我们更好的理解程序的内容，可以帮助软件开发人员、质量保证人员查找代码中存在的结构性错误、安全漏洞等问题，从而保证软件的整体质量。还可以用于帮助软件开发人员快速理解文档残缺的大规模软件系统以及系统业务逻辑抽取等系统文档化等领域。 如开发20年以上的金融核心COBOL系统，动辄上千万行代码的系统规模。对于理解这样规模的系统，基于程序静态分析的辅助理解工具就能发挥积极作用。因此本文将构造程序的控制流图作为课题研究的主要内容。

1. 国内外研究现状

2.1 国外研究现状

M.Weiser 1979年在他的博士论文[1]中提出程序切片技术，程序切片技术是优先建立起来的一种程序分析技术，1984年，M.Weiser在IEEE Transations on Software Engineering上进一步阐述了程序切片技术的思想[2]。程序切片技术在软件分析、理解、调试、测试、度量、 软件质量保证、逆向工程等许多方面有着广泛的应用。在软件调试过程中，定位程 序中存在的错误是一项困难的工作，程序切片可以帮助程序员很容易地进行错误定位。

Ferrante[3]在1987年提出了控制流图的概念，后来随着网络技术的发展和大量软件的使用，软件的安全越来越受到人们的关注，软件分析和测试成为软件开发过程中一个极为重要的环节，而基于控制流图的程序分析是软件安全分析的重要内容。控制流图中的节点分为两种类型[4]：一种是块结构，即把程序理解为块，块是一组连续的程序语句，块中只包含简单语句，不包含分支和循环等引起程序。另一种是程序的每一行代码都单独的看为一个控制流结点，这种结构也是目前使用最广泛的一种结构。从目前掌握的参考文献来看，控制流图的研究主要集中在控制流图的构造以及对控制流分析结果的显示。加拿大麦吉尔大学的R.V.Rai等人开发的java字节码分析工具Soot[5]，是比较早的程序代码分析工具，并且主要针对Java程序进行分析，它通过分析编译生成的.class文件可以自动的生成程序的控制流图，并把控制流图以图形化的方式展现给用户，然后在此基础上可以对程序进行分析和编译优化。佐治亚理工学院的G.Rothermel[6]等人对软件分析的各种技术进行了介绍，包括控制流图、数据流分析和函数依赖，并利用边界分析技术设计了从程序中得到基本程序“块”的算法，最后又设计了程序控制流图的生成算法，但是此算法无法对循环语句之间的嵌套进行分析。新加坡国立大学的Joxan Jaffar[7]等人提出了路径敏感的控制流图生成算法，此算法更关注程序在实际的运行中可能的执行路径，并去除控制流图中存在的不可执行路径，此算法需要设计相应的测试用类动态的生成一些数据，以确定程序的不可执行路径，且需要人工判断，所以实现起来有一定的难度。德国因戈尔施塔特大学的Robert Gold[8]等人提出了控制流图的精简算法。，其主要目的是在保证程序语句和分支的覆盖率的前提下，减少控制流图中的结点数目，从而更有利于进行程序分析，但是此方法没有对程序的语义进行相应的分析。奥地利林茨大学的Würthinger T对java程序的控制流图进行了可视化的研究[9]，并通过在 ECLiPSe[10]开发环境下开发相应的插件来实现控制流图的可视化输出。

2.2 国内研究现状

相对于国外，国内在控制流图的研究上也取得了一定的进展。武汉大学软件工程研究所的毋国庆、峋培基 周国强三人在1991年提出了一种把源程序转换为结构化的流程图（PAD图）的方法和实现技术[11]，该方法和技术已被用于转换工具IPADT系统。中国科学院计算技术研究所的杜子徳[12]对控制流图进行了研究，论述了控制流图（CFG）的实现途径以及使用方法。哈尔滨科学技术大学的吕励针对软件工程领域中逆向工程的研究课题，提出了一种利用图式把C语言源程序转换成结构化流程图（PAD图）的方法和技术[13]。孙永新[14]等人对程序的各种控制结构进行了深入的分析和研究，并在此基础上设计了一个实用的程序控制流图生成工具。

2.3 有待进一步解决的问题

通过分析国内外的研究状况，我们还有还有问题有待于进一步的解决。目前国内外对于控制流图的研究主要集中在程序的控制流图构造算法和控制流图的可视化研究上，且绝大多数都是针对java程序的，并利用ECLiPSe插件可以轻松的实现控制流图的可视化。针对C语言的研究则相对较少，虽然针对控制流图的生成算法进行了相关的研究，但在C语言控制流图的可视化上还存在很大的欠缺，可视化对图论知识的要求较高，实现起来有一定难度。

控制流图的可视化需要分析复杂的布图算法，且大部分布图算法只能画直线，所以当程序的逻辑很复杂的时候必将影响其效果。

1. 研究内容与方法

3.1 主要研究内容

（1） 对C语言的各种语句进行深入的分析和研究，弄清楚每种语句结构的语法规则和常见用法，为各种语句结构设计相应的控制流图生成算法。

（2） 根据控制流图生成算法，在语法分析规则中的相应位置添加语义动作，利用ANTLR自动生成工具自动的生成其所对应的控制流分析器，然后利用生成的控制流分析器对给定的C源程序进行分析便可得到逻辑表示。

（3）利用ECLiPse自动生成控制流图，并用ECLiPse提供的API进行可视化。

3.2 主要研究方法

3.2.1 控制流图概念及分析基本流程

程序内部元素指过程、基本块、语句、变量等。程序内部各元素的相互关系指程序调用图（即过程之何的相互调用关系）、控制流图、数据依赖图等等。从用户的角度看，他们往往着眼于不同的程序粒度，如过程级、基本块级、语句级和变量级等。程序调用图是粗粒度的控制流图，控制流图是中粒度的，而变量则是细粒度的。本文专门论述中粒度分析工具——控制流图。

控制流图是表示一个过程内所有基本块执行的可能流向和每个基本块所对应的语句表; 一个动态控制流图则能反映一个过程（subroutine）的实时执行过程。基本块（segment）是一个程序内可以连续执行的最大单位或对程序控制的最小单位，它的特点是单入口，单出口，中间没有分支。一个控制流图是对一个过程而言，不同的过程有不同的控制流图。在一个控制流图中，一个节点对应一个基本块，它对应控制流图的基本单位。对于每一个基本块，有一组语句（语句表）与其对应。

控制流图能反映出一个过程的信息包括。

1. 是哪一个过程；
2. 一个过程的入口（第一个基本块）和出口（最后一个基本块）；
3. 一个基本块的所有可能的下一个基本块（左右的出口）；
4. 一个基本块的所有可能的上一个基本块（所有的入口）；
5. 一个基本块所对应的语句表。

控制流分析以C程序源代码为输入，然后利用生成的控制流分析器对其进 行相应的分析，最后得到以图形化的方式表示的控制流图。其分析的基本流程如

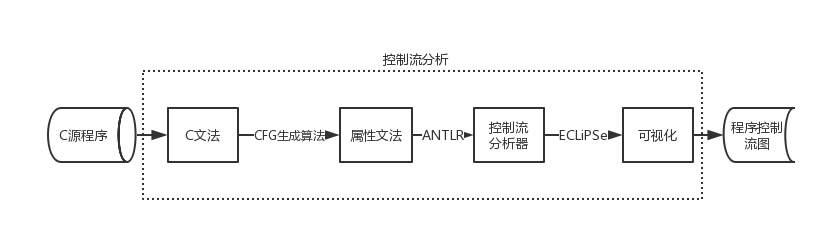
图1所示。

图1 控制流图分析基本流程图

图3.1展示了本文生成控制流图的基本过程。首先需要为C程序制定相应的 文法，然后根据C语言的语法特性，为各种C语句结构设计相应的控制流图生成算法，接下来根据控制流图生成算法在C文法中嵌入相应的语义动作，也即制定属性文法，然后利用分析器自动生成工具ANTLR生成控制流分析器。接下来使用开源的图表可视化软件Graphvizt301对控制流图进行可视化输出，利用ECLiPSe可以很好的解决控制流图的布图问题，使得控制流图的可视化效果更加直观。实验的最后我们把生成的控制流图以可视化的形式展现给用户。

3.2.2 控制流图生成算法的设计与实现

控制流图生成算法是构造程序控制流图的基础，当控制流分析器分析到一条 语句结点时，就调用相应的控制流图生成算法进行处理。在分析的过程中我们需

要用到语法制导翻译技术，其具体做法是在对应的文法中嵌入相应的语义动作， 其中的语义动作是相应控制流图生成算法的实现 。

3.2.3 C语言文法的分析

在语言的设计或编译器的设计过程中，文法是其中最主要的环节，文法给出 了精确的、易于理解的语法规范定义，它对于生成准确高效的分析器起着至关重 要的作用，此外结构清晰设计良好的文法还很容易根据实际的需要对文法进行扩 充。

本文使用旧金山大学的Terence Parr和DavidWigg专门针对ANTLR编写的 ANSIC词法分析规则和语法分析规则作为本研究的基本文法。其中词法分析 文法中定义的产生式的首字母必须大写，这些词法产生式可以识别所有的C语 言关键字，而所有的语法分析文法中定义的产生式的首字母必须小写，在词法分 析器所识别出的各个单词的基础上，这些语法产生式可以识别出C程序中的各 种语句结构。

3.2.4 控制流图的可视化

利用ECLiPse自动生成控制流图，并用ECLiPse提供的API进行可视化。

3.3 研究基础与条件

3.4 研究重点与难点

3.5 初步解决方案

1. 预期成果与进度安排

4.1 课题设计目标

本课题拟研究一个基于约束逻辑编程的C控制流图自动生成方法，设计并且实现将C语言程序自动生成程序控制流图的工具。预计主要工作是由三个部分组成：设计控制流图生成算法，语义动作的编写，控制流图的可视化，实现源程序的程序控制流图。

4.2 预期研究成果

课题研究的预期成果包括：

1. 一篇基于约束逻辑编程的C语言程序控制流图自动生成研究的论文

（2） 一个完整的能够将C语言程序生成程序控制流图的系统。设计从源程序自动生成其控制流图的算法，并利用ECLiPSe对控制流图进行了可视化输出；最后从理论上和实践上详细论述该系统的本科毕业论文将作为我的毕业设计的最终成果。

4.3 计划进度安排

毕业设计工作的时间进度安排如表1所示。

表1 毕业设计时间进度安排

|  |  |  |
| --- | --- | --- |
| **任务编号** | **起止时间** | **工作内容** |
| Task1 | 2017年2月19日~3月5日 | 完成对毕业设计课题的调研报告，对课题的研究方法有一定的了解，设计课题研究的基本思路和方法 |
| Task2 | 2017年3月6日~4月1日 | 相关英文文献的翻译，完成源程序生成程序控制流图的设计与编程实现 |
| Task3 | 2017年4月2日~4月20日 | 完成毕业论文初稿的撰写并交予老师审阅 |
| Task4 | 2017年4月21日~5月5日 | 根据论文定稿格式完成论文定稿 |
| Task5 | 2017年5月9日~5月20日 | 毕业论文修改，准备毕业答辩 |
| Task6 | 2017年5月20日~6月10日 | 完成答辩工作 |

1. 结论

本文在充分研究了C语言语法结构的基础上，设计了从源程序自动生成其程序控制流图的算法，并对程序控制流图进行了可视化输出。

首先，需要对程序控制流图的各种表现形式进行研究，明确各种语句结构的流程，接下来根据各种语句的流程设计了相应的数据结构和程序控制流图生成算法。其次，认真分析了ANTLR提供的ANSI C文法结构，并在文法中找到各种语句结构所对应的产生式，分析各产生式之间的相互关系，只有明确了各产生式之间的关系，我们才能够准确的嵌入相应的语义动作。接下来，根据程序控制流图生成算法编写相应的语义动作，并嵌入到文法中的相应位置，来实现程序控制流图的构造。最后，对程序控制流图进行可视化输出。

参考文献

[1] M.Weiser. Program slices: formal, psychological, and practical investigations of an automatic program abstraction method[M]. University of Michigan, 1979.

[2] Weiser M. Program Slicing[J]. IEEE Transactions on Software Engineering, 1984, SE-10(4):352-357.

[3] Ferrante J, Ottenstein K J, Warren J D. The program dependence graph and its use in optimization[M]. ACM, 1987.

[4] Heer J, Agrawala M. Software design patterns for information visualization[J]. IEEE Transactions on Visualization & Computer Graphics, 2006, 12(5):853-860.

[5] Valléerai R, Co P, Gagnon E, et al. Soot - a Java bytecode optimization framework[J]. Masters Abstracts International, Volume: 41-01, page: 0255.;Adviser: Laurie Hendren. 2010:214-224.

[6] Rothermel G. Representation and Analysis of Software[J]. Angewandte Chemie, 2005, 46(31):5896-900.

[7] Jaffar J, Navas J A, Santosa A E. A Path-Sensitive Control Flow Graph[J]. Comp.nus.edu.sg, 2011.

[8] Gold R. Control flow graphs and code coverage[M]. Versita, 2010.

[9] Würthinger T.Visualization of Java Control Flow Graphs.Bachelor thesis，Institute for System Software，Johannes Kepler University Linz，2006.

[10] Wallace. Constraint logic programming using ECLiPSe[M]. Cambridge University Press, 2007.

[11] 毋国庆, 陶培基. 一种源程序到流程图的转换方法及实现[J]. 软件学报, 1991(3):36-43.

[12] 杜子德. 程序控制流图:一种可观化的程序设计工具[J]. 计算机研究与发展, 1995(12):15-20.

[13] 吕励. C语言源程序转换为结构流程图的方法[J]. 哈尔滨理工大学学报, 1995(6):69-73.

[14] 孙永新, 吴家培, 闫大顺. 基于基本块标识方法的控制流图生成器设计[J]. 计算机应用与软件, 2010, 27(5):158-161.