Journal of Computer Applications
计算机应用,2013,33(12):3428 -3431
ISSN 1001-9081
CODEN JYIIDU
2013-12-01
http://www.joca.cn

文章编号:1001-9081(2013)12-3428-04
doi:10.11772/j.issn.1001-9081.2013.12.3428

基于结构分析的高级语言控制结构恢复方法

霍元宏*,刘　毅,计卫星
(北京理工大学 计算机学院,北京 100081)
(*通信作者电子邮箱 yuanhonghuo.439@163.com)

摘　要:为正确获得嵌入式可执行程序和汇编代码的高级语言控制结构,弥补现有高级程序控制结构恢复算法在处理非结构化区域的不足,将编译领域经典的控制流分析方法——结构分析算法引入到嵌入式汇编代码高级程序控制结构恢复研究中;针对嵌入式可执行程序的特点,对结构分析算法加以改进;利用结构分析算法的结果构造程序的控制树,生成高级语言代码。与开源反编译器 DCC 的对比实验结果表明,改进的结构分析算法在高级程序结构恢复问题上是可行有效的。

关键词:反编译;控制流分析;嵌入式系统;逆向分析
中图分类号:TP314　　**文献标志码**:A

Recovery method for high-level language control structures based on structural analysis

HUO Yuanhong*, LIU Yi, JI Weixing
(*School of Computer Science, Beijing Institute of Technology, Beijing* 100081, *China*)

Abstract: To correctly obtain the high-level language control structures of embedded executables and assembly code, and resolve the problem that the existing recovery methods for high-level language control structures cannot handle the unstructured region, the classical control analysis method, structural analysis algorithm, was introduced to study the recovery method for high-level control structures of embedded assembly code. The structural analysis algorithm was improved according to the characteristics of embedded executables, and the high-level language code was generated by using the program control tree, which can be obtained from the results of structural analysis algorithm. Compared with the open source decompiler named DCC, the results show that the improved algorithm is feasible and efficient.

Key words: decompiling; control flow analysis; embedded system; reverse analysis

0　引言

反编译是一个由机器代码或汇编代码生成更易理解的高级语言程序的过程。在这一过程中,根据程序控制流图提取高级程序控制结构是至关重要的一步。在嵌入式程序中,为减少生成的代码所占用的存储空间,编译器对程序进行了大量的优化,导致程序中的跳转错综复杂,提取出的控制流图中包含了大量的非结构化子图,对高级程序控制结构恢复造成了较大的影响。

目前控制结构恢复技术的研究主要针对桌面系统中的可执行程序进行,且主要关注结构化子图的分析与识别[1-3],这些方法能够提取出控制流图中的大部分控制结构,但是对于非结构化区域的处理略显不足。对于非结构化流图结构恢复问题的研究目前主要采取 4 种方法,分别是引入布尔变量[4-6]、代码复制[7-8]、引入特殊控制结构[9-10]和图转换系统[6,11]。采用布尔变量和代码复制的方法虽然可以获得等价的结构化的控制流图,但是改变了程序的语法和逻辑结构,引入特殊控制结构导致某些结构无法被大多数高级程序语言正确描述,图转化系统保持程序控制结构不发生变化,但是在高级代码生成阶段产生大量的"goto"语句。

为了更加准确地恢复嵌入式代码的高级程序控制结构,本文将编译领域经典的控制流分析算法——结构分析方法引入到控制结构恢复过程中,并在该方法上作了适当改进以适应嵌入式代码的特点。在对非结构化流图进行结构化的过程中,本文针对结构分析涉及的三类非结构化区域分别提出了基于规则的结构化方法。实验结果表明改进的结构分析算法能够更加准确地恢复程序的高级语言控制结构。

1　嵌入式代码高级程序控制结构恢复

结构分析是一种更为精致的区间分析方法,与基本的区间分析不同,它能表示出比循环更多的控制结构类型,并使每一种类型的控制结构形成一个区域。结构分析考虑流图的深度为主生成树,然后对各种区域类型的实例按后序次序依次考察流图中的节点,将它们规约为抽象节点,蜕化掉连接边,并构造对应的控制树[12]。

经典的结构分析算法可以准确识别顺序结构(图 1(a))、if-then(图 1(b))、if-then-else(图 1(c))、switch(图 1(e))、self-loop(图 1(f))和 while-loop(图 1(d))等六种结构化区域;非正常选择路径(图 2(a))、自然循环(图 2(b))和非正常区间(图 2(c))三种非结构化区域。需要说明的是,图 2 中的自然循环是指除了 self-loop 和 while-loop 以外的单入口循环。非正常选择路径区间是指既没有包含环路也不能规约为任何简单无环结构的区域,这种结构在嵌入式代码中很常见。图中所给出非结构化区域都是示意性的,因为循环可能不止两条出口边,非正常区间的入口基本块可能有两个以上的后继。

收稿日期:2013-08-05;**修回日期**:2013-08-14。

作者简介:霍元宏(1988 -),男,河南信阳人,博士研究生,主要研究方向:编译与反编译;　刘毅(1987 -),男,黑龙江鸡西人,硕士研究生,主要研究方向:计算机系统结构;　计卫星(1980 -),男,陕西咸阳人,讲师,主要研究方向:计算机系统结构、编译与反编译。

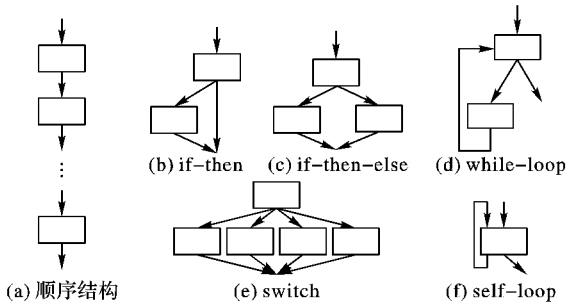

图 1　结构分析识别的结构化区域

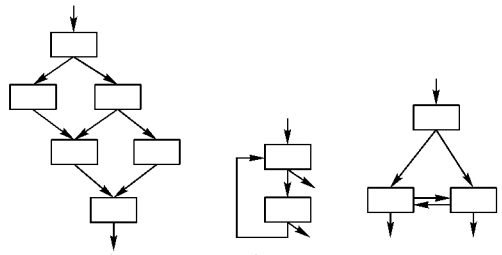

图 2　结构分析识别的非结构化区域

由于结构分析算法在编译领域主要用于控制流分析和数据流分析,它没有对非结构化区域进行结构化处理,也不适合直接用于嵌入式代码的结构恢复。为了使结构分析算法可以进行嵌入式代码的结构恢复,本文对算法进行了以下改进:

首先,由于在嵌入式汇编代码中,同一个函数中可能包含多条返回指令,这样得到的控制流图可能存在多个节点后继为空。在此情况下,如果严格按照结构分析算法,流图无法规约为一个节点。为了让结构分析算法顺利进行,本文将结构分析算法可以识别的结构加以扩充,使得其可以识别如图 3 所示的结构。图中的虚线表示该边可以不存在,这样较好地解决了一个函数包含多条返回指令的问题。

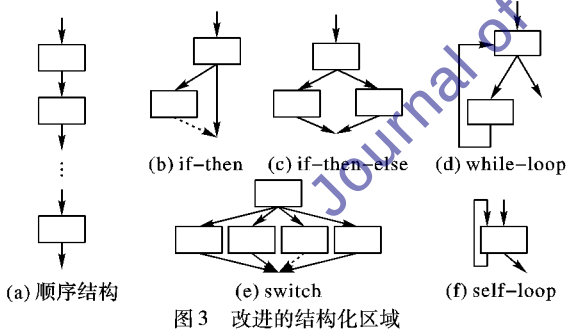

图 3　改进的结构化区域

其次,结构分析算法识别到非结构化区域时,并不对非结构化区域进行结构化处理,而非结构化区域无法用现有的高级语言控制结构表示。为了解决这个问题,本文设计了算法对这些非结构化区域进行结构化处理。该算法对非结构化流图进行剪边处理,使其结构化。剪下的边将被记录,在高级代码生成阶段,产生相应的“goto”语句形式。对结构分析得到的三种非结构化区域的处理规则定义如下:

非正常选择路径区间无法规约,是由于 if-then-else 结构的一个分支有多个前继,所以,只需要把通向该分支的边剪下,便可以将流图结构化。对非正常选择路径区间的处理方式如图 4 所示。

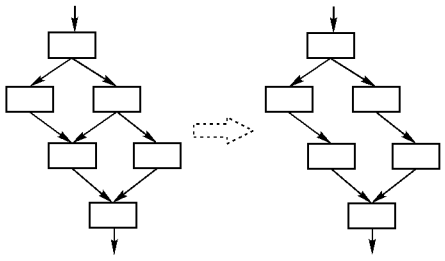
图 4　非正常选择路径区间的结构化

导致自然循环无法规约的原因是循环有多个出口,所以结构化方法可以选择破坏循环结构(图 5(a))或者剪掉多余的出口(图 5(b))。在具体对非结构化流图进行处理时,可以根据程序的上下文选择合适的处理方式。

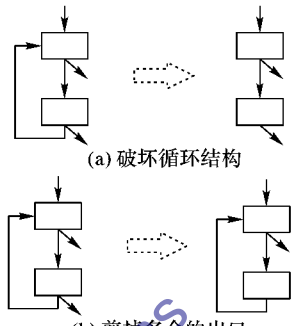

图 5　自然循环的结构化

导致非正常区间无法规约的原因在于它包含一个循环,该循环有多个入口,所以非正常区间结构化的方法可以选择剪掉多余的入口(图 6(a))或者破坏循环(图 6(b)),需要根据程序的上下文选择更合适的处理方案。

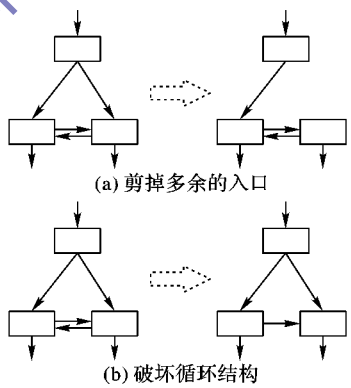

图 6　非正常区间的结构化

需要说明的是,对于非结构化区域的处理,往往不能简单通过一步处理实现结构化。在实际处理过程中,可以按以上规则先剪一条边,然后对流图进行规约,如果不能完全规约,则对非结构化区域继续结构化处理,直到程序流图可以被完全规约。

经过以上改进,结构分析算法便可以更好地应用于嵌入式代码的高级程序控制结构恢复当中。下面通过一个控制流图的处理过程,详细介绍本文算法的工作流程。

图 7(a)是一个非结构化程序的控制流图,它包含 7 个基本块,分别记为 B1 ~ B7,在进行控制流分析时,其执行过程如下:

步骤 1　对控制流图进行深度优先后序遍历,获得控制流图的后序遍历序列:7, 5, 6, 3, 4, 2, 1。

步骤 2　依次对后序序列中的节点进行分析,判断其是否符合某一种高级程序控制结构,当扫描到基本块 B6 时,B6 包含两个后继却不能识别为 if-then 或者 if-then-else,所以它被认定是非结构化区域中的点,然后查找该非结构化区域包含的节点,可以找到 B3、B5、B6、B7 四个基本块构成非正常选择路径区间结构。

步骤 3　对找到的非结构化区域进行结构化处理，利用前文定义的剪边规则，将流图中的边 B6→B5 剪去，得到图 7(b)。

步骤 4　由于控制流图已经改变，重新扫描后续遍历序列进行规约，当遍历到基本块 B3 时，B3 有两个后继 B5、B6，且符合 if-then-else 结构特点，所以节点 B3、B5、B6 被抽象为一个新的节点，记为 B8，控制流图也被变换为图 7(c)。

步骤 5　继续对后续遍历序列进行扫描，到达节点 B2，B2 有两个后继，两个前驱，并且 B4 和 B2 构成一个环路，B2 和 B4 符合 while 循环特点，所以可以进行规约，抽象得到节点 B9，控制流图转换为图 7(d)。

步骤 6　继续对后续遍历序列进行扫描，到达基本块 B1，B1 是入口基本块，且只有一个后继，不断沿着其后继进行分析可以知道基本块 B1、B9、B8、B7 满足顺序结构特点，所以可以规约得到节点 B10，此时，控制流图变为图 7(e)。由于控制流图已经抽象成一个节点，说明控制流分析已经完成。

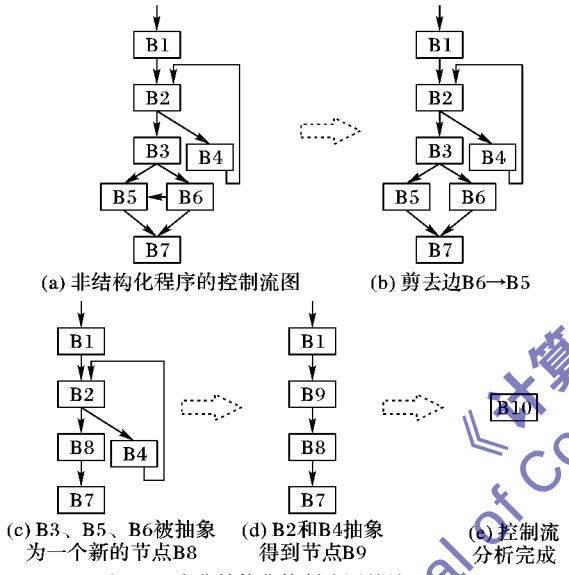

图 7　一个非结构化控制流图的处理流程

经过控制流分析后，可以得到程序的控制树，如图 8 所示。

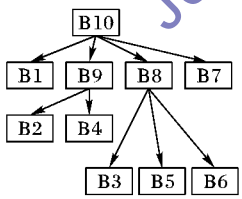

图 8　结构分析后得到的控制树

在图 8 所示的控制树中，叶子节点是代码划分得到的基本块编号，其他节点是结构分析得到的控制结构信息。需要说明的是，控制树展现的是结构化的程序信息，对于非结构化控制流图，无法展示剪下的边的信息，剪下的边被另外保存，在代码生成阶段进行特别处理。

2　高级代码生成

经过结构分析算法处理后，程序的控制流图便被转换为一棵控制树和一个被剪的边的集合(对于非结构化流图)，这棵控制树的每个节点包含以下属性：节点 ID、结构类型和所包含子节点信息等，通过这些信息，便可以进行程序的高级代码生成，高级代码生成算法如下：

输入：节点 ID；
输出：该节点对应的高级代码形式。

```
GenCode( NodeID)
{
  根据节点 ID 找到相应节点 CurrNode;
  如果节点 CurrNode 的类型是基本块
    进行该基本块的高级代码生成;
    如果该基本块的后继有剪边的情况,
      查找剪下的边的详细信息;
      将该边的后继基本块在高级代码生成时加上标号;
      该基本块加上跳转到后继基本块的"goto"语句;
  如果节点 CurrNode 类型是高级程序控制结构
    根据控制结构类型,产生高级代码框架;
    递归调用高级代码生成函数,对孩子节点进行高级代码生
    成,将生成的高级代码嵌入到代码框架中;
  返回生成的高级代码
}
```

该算法的特点是严格按照结构分析得到的程序控制结构信息进行代码生成，对于剪下的边进行“goto”语句的生成。由于结构分析算法完整地提取了程序的控制结构，而代码生成算法也完整地展现了结构分析的结果，从而保证最终生成的高级代码完全体现了汇编代码的结构信息。

下面针对上文提到的例子详细描述该算法的执行过程：

从结构 B10 开始进行代码生成，B10 包含结构 B1，B9，B8，B7 四个子结构。依次对这四个子结构进行高级代码生成。B1，B7 是基本块，首先进行基本块代码生成，然后判断 B1，B7 是否有剪边处理，若有，则进行“goto”语句生成处理。B9，B8 为非叶子节点，分别递归调用高级代码生成算法进行处理。

当进行基本块 B6 的高级代码生成时，B6 有剪边处理，所以需要额外产生一条“goto”语句，而对于基本块 B5，由于有通向 B5 的边被剪，所以 B5 需要产生一个标签。这样得到的高级代码既保证了与汇编代码的等价性，又最大可能地接近高级代码表示形式。本文的例子经过高级代码生成后得到的高级代码形式如下：

```
B1. code
while( B2. code)
   {B4. code}
if ( B3. code)
  {if( B6. code)  goto L1;}
else{L1:  B5. code}
B7. code
```

3　实验结果及分析

为了对本文的方法进行可行性和有效性验证，从 Intel 8051 可执行程序中随机选择了 10 个非结构化程序的控制流图，并利用基于本文所述方法设计实现的反编译器 BIT-DEC 对这些流图进行反编译。为了更好地说明本文的算法是有效的，利用一款开源的反编译工具 DCC 对同样的测试用例进行分析，并将实验结果加以对比。实验结果如表 1，其中“—”表示产生的高级代码没有正确描述汇编代码的结构。

通过对实验结果的分析可知，改进的结构分析算法可以完全正确地描述嵌入式代码的程序结构，而 DCC 方法有 4 个函数产生的高级代码没能正确反映汇编代码的程序结构和内容。由于在对非结构化区域结构化的过程中，采用的是基于规则的处理方式，因此无法保证在所有情况下都剪最少的边，进而产生最少的“goto”语句，这也是为什么图 1 和图 4 剪下

的边比 DCC 算法多的原因。

表 1　分析结果

图	边数	点数	剪下的边数	
			本文方法	DCC 方法
1	25	17	3	2
2	30	21	2	—
3	10	7	1	1
4	28	19	3	2
5	53	38	3	—
6	39	30	2	6
7	34	27	1	2
8	33	22	3	—
9	24	16	2	3
10	15	11	2	—

4　结语

本文在深入了解嵌入式汇编代码程序结构的基础上,分析了现有高级程序控制结构恢复算法的不足,提出了利用编译领域经典的结构分析算法进行嵌入式代码高级程序控制结构恢复的方法,在实现的过程中,针对嵌入式代码的特点,对经典结构分析算法进行了改进,并针对结构分析算法没有很好解决的非结构化区域结构化的问题进行了深入研究,提出了可行的解决方案。最后,在结构分析结果的基础上,设计算法实现了高级代码的生成算法。通过与开源反编译器 DCC 的对比,实验结果表明,本文提出的算法是正确和有效的。

参考文献:

[1] 侯文永,徐志宏. 反编译过程中的结构变换[J]. 上海交通大学学报, 1996, 30(6): 81 – 84.

[2] 赵蕾,王开铸. C 反编译控制流恢复的形式描述及算法[J]. 计算机学报,1998,21(1):87 – 91.

[3] 刘宗田,兰群. C 子集程序到 C 语言程序的变换[J]. 计算机研究与发展,1991,28(3):29 – 34.

[4] BOHM C, JACOPINI G. Flow diagrams, turing machines and languages with only two formation rules [J]. Communications of the ACM, 1966, 9(5): 366 – 371.

[5] ASHCROFT E, MANNA Z. The translation of ‘go to’ programs to ‘while’ programs [M]// Classics in Software Engineering. Upper Saddle River: Yourdon Press, 1979: 49 – 61.

[6] EROSA A M, HENDREN L J. Taming control flow: a structured approach to eliminating goto statements [C]// Proceedings of the 1994 International Conference on Computer Languages. Piscataway: IEEE, 1994: 229 – 240.

[7] KNUTH D E, FLOYED R W. Notes on avoiding ‘go to’ statements [J]. Information Processing Letters, 1970, l(1):22 – 23

[8] OULSNAM G. Unravelling unstructured programs [J]. The Computer Journal, 1982, 25(3): 379 – 387.

[9] BAKER B S. An algorithm for structuring flowgraphs [J]. Journal of the ACM, 1977, 24(1): 98 – 120.

[10] CIFUENTES C, GOUGH K J. A methodology for decompilation [C]// Proceedings for the XIX Conferencia Latinoamericana de Informatica. Buenos Aires: [s. n.], 1993: 257 – 266.

[11] LICHTBLAU U. Decompilation of control structures by means of graph transformations [C]// CAAP ’85: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Volume 1: Colloquium on Trees in Algebra and Programming: Mathematical Foundations of Software, LNCS 185. Berlin: Springer-Verlag, 1985: 284 – 297.

[12] MUCHINICK S S. Advanced compiler design and implementation [M]. San Francisco: Morgan Kaufmann, 1997.

(上接第 3422 页)

今后,希望能通过从梯形逻辑到模型检测规范语言的自动转换,实现高效的自动化验证技术,最终目的是能在设计阶段较早地获得安全性能估计。

参考文献:

[1] 燕飞. 轨道交通列车运行控制系统的形式化建模和模型检验方法研究[D]. 北京:北京交通大学,2006:6 – 20.

[2] 唐涛,徐田华,赵琳. 列车运行控制系统规范建模与验证[M]. 北京:中国铁道出版社,2010:25 – 110.

[3] 边计年,薛宏熙,苏明,等. 数字系统设计自动化[M]. 北京:清华大学出版社,2005:327 – 380.

[4] HAXTHAUSEN A E. An introduction to formal methods for the development of safety-critical applications[D]. Lyngby: Technical University of Denmark, 2010:6 – 19.

[5] 宁宁,张骏,高向阳. 基于符号模型检测的符号有向图故障诊断解形式化验证[J]. 信息与控制,2010,39(4):423 – 429.

[6] CAVADA R, CIMATTI A. NuSMV 2.5 user manual [K/OL]. [2013 – 02 – 12]. http://nusmv.fbk.eu/NuSMV/userman/v25/nusmv.pdf.

[7] ERIKSEN L E. Verification of safety properties for relay interlocking systems [EB/OL]. [2013 – 02 – 16]. http://etd.dtu.dk/thesis/266717/ep10_57_net.pdf.

[8] JAMES P. SAT-based model checking and its applications to train control systems [EB/OL]. [2013 – 02 – 20]. http://www.cs.swan.ac.uk/~csmarkus/ProcessesAndData/Papers/james10a.pdf.

[9] JAMES P, ROGGENBACH M. Designing domain specific languages for verification: first steps [C/OL]// ATE’11: Proceedings of the 2011 Australian Tourism Exchang. 2011: 40 – 45. http://www.cs.swansea.ac.uk/~csmarkus/ProcessesAndData/Papers/james11a.pdf.

[10] JAMES P, ROGGENBACH M. Automatically verifying railway interlockings using SAT-based model checking [J]. Electronic Communications of the EASST, 2010, 35(2010):3 – 9.

[11] 燕飞,唐涛. 计算机联锁控制逻辑的模型检验方法[J]. 铁道通信信号,2009,45(5):26 – 29.

[12] LI J, CHEN S. Design of software based on railway transportation interlocking control testing system [J]. Railway Computer Application, 2011, 20(3):46 – 49.

[13] HEI X, OUYANG N. The scheduling strategy of concurrent request in distributed railway interlocking system [J]. ICIC Express Letters, Part B: Applications, 2011, 2(1):43 – 48.

[14] 张军林. NuSMV 模型验证器实现分析[D]. 广州:中山大学,2010: 19 – 23.

[15] HAXTHAUSEN A E. Automated generation of safety requirements from railway interlocking tables [C]// ISoLA’12: Proceedings of the 5th International Conference on Leveraging Applications of Formal Methods, Verification and Validation: Applications and Case Studies, LNCS 7610. Berlin: Springer-Verlag, 2012: 261 – 275.

[16] ZAFAR N A, KHAN S A, ARAKI K. Towards the safety properties of moving block railway interlocking system [J]. International Journal of Innovative Computing, Information and Control, 2012, 8(8): 5677 – 5690.