Journal of Computer Applications
计算机应用,2013,33(12):3419-3422,3431
ISSN 1001-9081
CODEN JYIIDU
2013-12-01
http://www.joca.cn

文章编号:1001-9081(2013)12-3419-04
doi:10.11772/j.issn.1001-9081.2013.12.3419

基于梯形逻辑的联锁系统形式化验证方法

于丽贞,徐中伟,陈祖希,张舒青*
(同济大学 电子与信息工程学院,上海 201804)
(*通信作者电子邮箱 zhang415535492@163.com)

摘　要:铁路联锁系统设计通常采用梯形逻辑进行建模。为了实现对铁路联锁系统进行形式化验证的目的,根据梯形逻辑的状态变迁语义,将梯形逻辑表示的联锁系统模型转换成模型检测工具 NuSMV 的语言,并将铁路联锁系统的安全需求表示为计算树逻辑(CTL),最后实现基于 NuSMV 的铁路联锁系统设计模型的形式化验证。

关键词:铁路联锁系统;模型检测;形式化方法;梯形逻辑;NuSMV 模型检测
中图分类号:TP311　　**文献标志码**:A

Formal verification of railway interlocking system based on ladder logic

YU Lizhen, XU Zhongwei, CHEN Zuxi, ZHANG Shuqing*
(*College of Electronics and Information Engineering, Tongji University, Shanghai* 201804, *China*)

Abstract: Ladder logic was used to model the railway interlocking system. In order to achieve the purpose of formal verification of the railway interlocking system, the model of the railway interlocking system expressed by ladder logic was converted to NuSMV language, which was a temporal logic model checker. Then Computational Tree Logic (CTL) specification representing the safety requirements of the railway interlocking system was verified. Finally, the formal verification of computer design model of the railway interlocking system was implemented based on NuSMV.

Key words: railway interlocking system; model checking; formal method; ladder logic; NuSMV model checking

0　引言

在现代公共交通体系中,轨道交通系统具有不可替代的突出地位。如何实现列车安全、快速、高效的运行,是摆在相关科研人员面前的一个突出问题。铁路车站计算机联锁系统是铁路行车的重要设备,它所包含的相互制约关系和控制顺序往往十分复杂,所以联锁控制逻辑设计的正确性直接关系到列车的运行安全。随着计算机技术在铁路信号系统的应用,安全问题显得越来越重要,传统的系统设计、分析和测试方法已经难以满足系统的安全需要。近年来,基于离散数学的形式化方法发展迅速,为解决安全计算机系统设计开发的正确性问题提供了一条可能的途径[1]。

作为一种图形化语言,由于其简单的逻辑关系、直观的表达方式,梯形逻辑在时序系统的设计领域得到广泛应用。采用梯形逻辑对联锁控制逻辑进行设计,可以将联锁系统表述为一个迁移系统,通过这种基于梯形逻辑的迁移系统可以得到系统内部的状态空间。而形式化验证方法就是对状态空间进行搜索,以期检测这个系统模型是否满足某些给定属性。

本文针对梯形逻辑建模的特点,研究形式化验证的理论,尝试采用形式化方法对基于梯形逻辑的联锁控制系统设计模型进行验证,选取了计算树逻辑(Computational Tree Logic, CTL)符号模型检测方法对系统的设计模型进行形式化验证,最后选取实例进行模型到代码转化的应用,探索出基于梯形逻辑设计模型形式化验证的一整套方法。

1　模型检测

模型检测是一种验证系统属性的形式化方法,它从数学上完备地验证系统实现是否与规范一致。模型检测通常采用状态空间遍历技术检测一个给定的计算模型是否满足用时态逻辑公式表示的特定属性。模型检测对验证计算机系统的正确性具有传统方法无法比拟的优势,如全自动进行而无须人机交互,定位设计错误,系统建模的完整性等,尤其是对安全苛求系统,这种优势更为明显[2]。

模型检测的主要思想是:将待检测的系统抽象为有限状态机,用时态逻辑描述系统应该满足的性质,然后遍历有限状态机的状态空间,对每个状态判断是否满足这些性质。若不满足,给出一个不满足性质的状态序列[3]。所以模型检测一般包含建模、规范、验证三个步骤。

属性是与系统行为发生次序有关的时态属性,通常用时态逻辑表示。时态逻辑是描述系统所处状态的性质以及状态间迁移序列的表达式,它可以方便准确地描述并发系统的重要性质,如安全性(safety)、活性(liveness)和公平性(fairness)。安全性是指系统的正确性和互斥性,用于说明"某条件永远满足,或某危险事件永不发生";活性是指系统的终止性、无活死锁性和响应性等,用于说明"某必需事件终将发生";公平性是指系统的合理性,用于说明"某事件必须无限经常地发生"[3]。CTL 是首个用于模型检测过程的时序逻辑语言。以一个有穷状态并发系统的 Kripke 结构的初始

收稿日期:2013-07-19;**修回日期**:2013-08-20。
基金项目:国家自然科学基金资助项目(60674004);国家 863 计划项目(2012AA112801)。
作者简介:于丽贞(1989-),女,陕西岐山人,硕士研究生,主要研究方向:形式化验证;　徐中伟(1964-),男,江苏无锡人,教授,主要研究方向:基于通信的列车控制系统与软件、通信协议的安全性形式验证和测试评估;　陈祖希(1981-),男,四川泸州人,博士研究生,主要研究方向:安全软件测试与评估;　张舒青(1989-),男,上海人,硕士研究生,主要研究方向:模型检测。

状态为树根,状态为节点,按各节点后继不同将 Kripke 结构展开,构成一棵计算树,树中任一条路径刻画系统的一次运行。

模型检测工具如 SMV、Spin 等,以模型和属性公式为参数,通过执行验证算法自动搜索整个状态空间,以验证模型是否满足属性[4]。算法的实现思想是通过穷举状态空间以计算出令公式为真的所有状态。

2 NuSMV 分析软件

符号模型检测(Symbolic Model Checking, SMC)是一种用布尔公式隐式的表示系统状态集合和迁移关系,并在符号状态空间上进行搜索的技术。对于要验证的 CTL 公式,通过对迁移关系进行相应的不动点运算,即可得到不动点的二叉决策图(Binary Decision Diagram, BDD)表示,用来进一步分析系统是否满足被验证性质[3]。下面给出 CTL 模型检测的伪代码:

```
Procedure SMC (M_BDD, f)
Begin
  将性质f转化为只包含 ¬、∧、EX、EU、EG 连接的表达式;
  OUT_BDD = Check(M_BDD, f);
  if OUT_BDD ∧ S_0BDD ≠ False
    return satisfy
  else
    return unsatisfy.
End SMC
```

SMC 过程中,子过程 Check 以有限状态机的 BDD 表示 MBDD 和 CTL 表达式 f 作为输入,输出结果是满足 f 的状态集合的 BDD 表示。S_{0BDD} 表示初始状态集合 S_0 的 BDD。

在符号模型检测方法的基础上,开发了模型检测工具 SMV,而 NuSMV 是 SMV 的重新实现和扩展,是第一款基于 BDD 的模型检测器。NuSMV 被设计成一个开放式的系统,不仅开源而且很容易修改、定制和扩展。它具有一个描述层次有穷状态并发系统的规范语言,并从系统规范中提取以 BDD 形式表示的迁移系统,然后用基于 BDD 的搜索算法确定系统是否满足 CTL 描述的被验证属性[6]。

NuSMV 分析软件以有限状态系统说明及其系统属性作为输入,若有限状态系统具有给定的属性,则输出 true,否则输出 false,同时提供一个违反规约的反例。反例可以在不同的冗余层次或以可重用数据结构的方式来产生,并且可以被检查和操纵。系统说明部分用 NuSMV 规定的 SMV 语言编程,属性部分则用 CTL 公式表达[7]。

3 梯形逻辑

梯形逻辑是一种图形化语言,由于其生成的梯形图呈阶梯状而得名,往往被用来对可编程逻辑控制器进行编程[8]。梯形图中,假想左右两侧母线(左母线和右母线)之间有一个左正右负的直流电源电压,母线之间有“能流”从左向右流动。依照梯形逻辑的这一特点设计系统模型,可以简化系统,直观地刻画出系统状态随时间顺序变换的迁移系统,实现对系统需求完全、正确和一致的描述。由于其简单的逻辑关系、直观的表达方式,梯形逻辑在时序系统的设计领域得到广泛应用[9]。

在梯形逻辑中,可以表示出两个状态变量集合:$V = \{v_0, v_1, \cdots, v_{n-1}\}$ 表示当前状态的集合,$V' = \{v_0', v_1', \cdots, v_{n-1}'\}$ 表示次态变量的集合。梯形图中的每一个梯级所表示的就是一次 $V \to V'$ 的前向遍历过程,即由现态求次态的过程。用 R 表示一个梯级的命题公式,则命题公式 $R \equiv v' \Leftrightarrow \Psi$ 表示当状态变量满足 Ψ 时,v 的下一状态改变。这样,就可以得到整个梯形逻辑的命题公式 $\Psi P \equiv ((v_1' \Leftrightarrow \Psi_1) \wedge (v_2' \Leftrightarrow \Psi_2) \wedge \cdots \wedge (v_n' \Leftrightarrow \Psi_n))$[9]。

梯形逻辑是一个不断迭代的状态迁移过程,可以用每次执行后变量的不同状态来对系统的行为进行建模,故而引入自动机理论方法来对系统进行建模[7]。用命题公式 Ψ_P 表示一个梯形逻辑,可以得到一个有限自动机

$$A(\Psi_P) = (S, I_S, \to)$$

其中:$S = \{\mu | \mu: I \cup C \to \{0,1\}\}$ 是一个状态集合,I 为输入变量集合,C 为梯形图梯级上变量的集合,μ 为一个布尔变量,$\mu \to \mu'$ 表示从 μ 到 μ' 的迁移;I_S 表示初始状态的集合。

对于一个给定的梯形逻辑图,可以得到一个有限自动机 $A(\Psi_P)$。如果存在一系列的迁移 $\mu_0 \to \mu_1 \to \cdots \to \mu \to \mu'$,$\mu_0 \in I$,那么说明从 μ 到 μ' 是可达的[10]。

4 基于梯形逻辑的联锁控制逻辑的模型检测

随着计算机技术的提高,计算机联锁系统变得越来越复杂,规模也越来越大。复杂的系统很难设计开发,寻找错误和安全隐患也比较困难,一般的测试方法很难发现系统所有的设计故障,这会给轨道交通系统的安全、高效运行带来一定的安全隐患。采用模型检测方法对车站联锁安全条件和联锁控制逻辑的设计进行验证,可以在很大程度上减少由系统开发人员造成的设计故障,提高系统的安全性。

4.1 用梯形逻辑设计联锁逻辑

联锁控制逻辑包括 6 个模块:操作命令形成模块、操作命令执行模块、进路处理模块、状态输入模块、表示输出模块以及控制命令输出模块。其中,进路处理模块是联锁控制逻辑的核心,它的主要工作是在执行了进路搜索子模块对所办进路以形成进路表之后,对进路进行处理[11]。梯形逻辑图可以很直观地表示出这种复杂的联锁逻辑关系,故如图 1 所示,选择一个简单的车站联锁站场图用梯形逻辑进行建模。

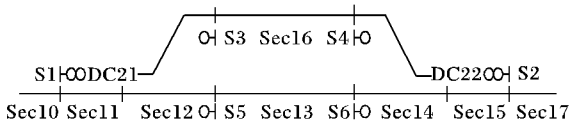

图 1 车站连锁站场图

对联锁控制逻辑的设计如下:进路操作输入后,进路搜索程序从站场数据库中选出进路,执行进路处理模块,共分为 5 个模块:区段检查和征用模块、道岔控制命令生成模块、进路锁闭模块、信号开放及关闭模块、区段解锁模块[12]。

1)区段检查和征用模块。要办理一条进路,首先检查该进路上所有区段是否空闲,是否已被征用。如果该进路上所有区段都空闲,并且未被征用,则征用这些区段。

2)道岔控制命令生成模块。道岔有定位和反位两个工作位置,只有当道岔所在区段未锁闭时,道岔才能改变其工作位置。道岔位置与进路以及进路始端信号机间的联锁关系如表 1 所示。以道岔 21 为例,得到道岔转换的梯形图如图 2 所示。

3)进路锁闭模块。为了保证行车安全,信号机开放时,首先必须把进路中的所有区段置成锁闭状态,且把敌对进路锁闭在未建立状态,这种锁闭叫进路锁闭。首先检查进路中

的所有区段空闲、道岔位置正确,并且进路中的所有区段被该进路征用,当这些条件全部满足时,实现进路的锁闭。

4)信号机开放及关闭模块。在信号开放时需要满足以下技术条件:进路中道岔位置正确且道岔区段锁闭、进路中的所有区段空闲、敌对进路锁在未建立状态[13]。列车一旦驶入进路,立即关闭信号机。

5)区段解锁模块。当进路锁闭后,进路上的所有区段实现锁闭。实现三点检查法解锁区段:列车出清上一区段并解锁;列车出清本区段,本区段空闲;且在列车进入下一区段,下一区段被占用时,本区段解锁。以区段 12 为例进行说明,得到梯形图如图 3 所示。

表 1　道岔与进路以及信号灯间的联锁关系

进路号	进路名称	道岔位置	始端信号
1	下行侧线接车	(21)	S1
2	下行正线接车	21	S1
3	上行侧线发车	(21)	S3
4	上行正线发车	21	S5
5	下行侧线发车	(22)	S4
6	下行正线发车	22	S6
7	上行侧线接车	(22)	S2
8	上行正线接车	22	S2

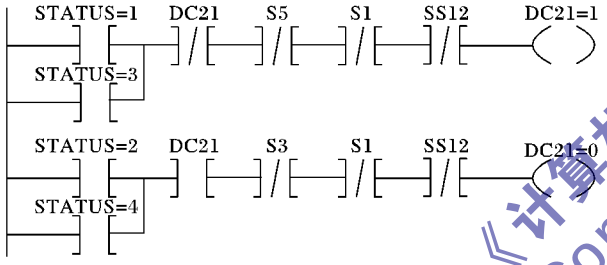

图 2　道岔 21 转换梯形图

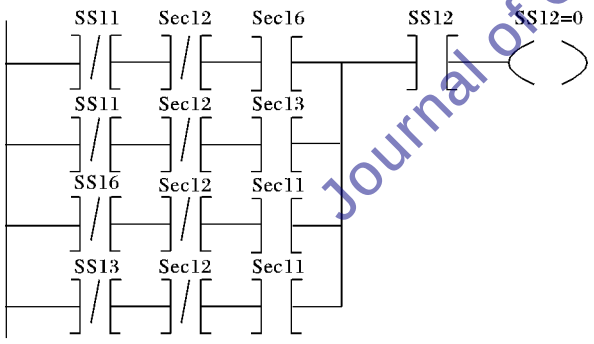

图 3　区段 12 解锁梯形图

4.2　梯形逻辑的模型描述

梯形图的状态遍历特点与形式化验证的状态遍历特点不谋而合,在这里使用模型检测方法对梯形图进行验证。首先采用 NuSMV 分析软件对用梯形图表示的系统进行建模,然后将道岔、进路、信号机之间的连锁关系用 CTL 表示出来,经过 NuSMV 分析软件的符号模型检测模块,可以确定基于梯形逻辑的设计模型是否符合计算机联锁控制逻辑的逻辑关系。

NuSMV 有两个很有用的表达式:init 表达式和 next 表达式。init 表达式用于描述初始状态,next 表达式用于描述转移关系。NuSMV 输入语言写的程序常被称为 smv 程序。所以在将梯形图模型转化为 smv 程序时:首先将梯形图中涉及的状态变量进行声明;然后就可以将梯形图的每一个梯级描述成为一个 next 表达式,即当同一梯级的能流所经过的元件对应的变量状态满足时,输出元件对应变量的下一状态改为相应状态。这样,利用 NuSMV 语言中的 next 表达式可以准确地描述梯形图表示的有限状态系统的逻辑关系,生成状态变迁关系的模型[14]。

根据图 2,可以将道岔 21 工作位置转换的梯形逻辑转化为 smv 代码如下:

```
next(DC21) : =
case
  DC21 = 0:
  case
    !(S1 = 1) &!(S5 = 1) &(STATUS = 1 | STATUS = 3) &SS12 = 0: 1;
    1: 0;
  esac;
  DC21 = 1:
  case
    !(S1 = 1) &!(S3 = 1) &(STATUS = 2 | STATUS = 4) &SS12 = 0: 0;
    1: 1;
  esac;
esac;
```

已知图 3 为用三点检查法解锁区段 12 的梯形逻辑表示,将其转化为 smv 代码:

```
next(SS12) : =
case
  SS12 = 1:
  case
    SS11 = 0&Sec12 = 0&Sec16 = 1: 0;
    SS11 = 0&Sec12 = 0&Sec13 = 1: 0;
    SS16 = 0&Sec12 = 0&Sec11 = 1: 0;
    SS13 = 0&Sec12 = 0&Sec11 = 1: 0;
    1: 1;
  esac;
  SS12 = 0:
  case
    JS1 = 1 | JS2 = 1 | JS3 = 1 | JS4 = 1: 1;
    1: 0;
  esac;
esac;
```

4.3　梯形逻辑的模型检测

4.3.1　安全属性 CTL 表达

联锁控制逻辑的安全性是指系统的正确性和斥性,用于说明“坏事情永远都不会发生”[15],主要有以下两种情况:

1)道岔与进路之间的联锁关系。

道岔与进路之间的联锁关系在进路选排时进行条件设定。在获取安全性时,可以用道岔与信号机之间的联锁关系表示[16],见表 2。

表 2　道岔与信号机之间的联锁关系表

信号机	进路号	道岔位置
S1	1、2	21、(21)
S2	7、8	22、(22)
S3	3	(21)
S4	5	(22)
S5	4	21
S6	6	22

用 CTL 公式表示只有道岔位置正确,信号机才能开放:

AG!(S5 = 1&DC21 = 1)(永远不会存在信号机 S5 开放,而道岔 21 处于反位的情况)

AG!(S3 = 1&DC21 = 0)(永远不会存在信号机 S3 开放,而道岔 21 处于定位的情况)

AG!（S6 = 1&DC22 = 1）（永远不会存在信号机 S6 开放，而道岔 22 处于反位的情况）

AG!（S4 = 1&DC22 = 0）（永远不会存在信号机 S4 开放，而道岔 22 处于定位的情况

2）进路与进路之间的联锁关系。

办理进路时，需要占用同一区段的进路为敌对进路，进路间的敌对关系见表 3。

表 3 进路与进路之间的联锁关系表

进路号	进路名称	敌对进路
1	下行侧线接车	2、3、4、7
2	下行正线接车	1、3、4、8
3	上行侧线发车	1、2、4
4	上行正线发车	1、2、3
5	下行侧线发车	6、7、8
6	下行正线发车	5、7、8
7	上行侧线接车	1、5、6、8
8	上行正线接车	2、5、6、7

CTL 公式表示敌对进路不能同时建立：

AG!（JS1 = 1 & JS2 = 1）（永远不会出现进路 1 和进路 2 同时建立的情况）

AG!（JS1 = 1 & JS3 = 1）（永远不会出现进路 1 和进路 3 同时建立的情况）

AG!（JS1 = 1 & JS4 = 1）（永远不会出现进路 1 和进路 4 同时建立的情况）

AG!（JS1 = 1 & JS7 = 1）（永远不会出现进路 1 和进路 7 同时建立的情况）

AG!（JS2 = 1 & JS3 = 1）（永远不会出现进路 2 和进路 3 同时建立的情况）

AG!（JS2 = 1 & JS4 = 1）（永远不会出现进路 2 和进路 4 同时建立的情况）

AG!（JS2 = 1 & JS8 = 1）（永远不会出现进路 2 和进路 8 同时建立的情况）

AG!（JS3 = 1 & JS4 = 1）（永远不会出现进路 3 和进路 4 同时建立的情况）

AG!（JS5 = 1 & JS6 = 1）（永远不会出现进路 5 和进路 6 同时建立的情况）

AG!（JS5 = 1 & JS7 = 1）（永远不会出现进路 5 和进路 7 同时建立的情况）

AG!（JS5 = 1 & JS8 = 1）（永远不会出现进路 5 和进路 8 同时建立的情况）

AG!（JS6 = 1 & JS7 = 1）（永远不会出现进路 6 和进路 7 同时建立的情况）

AG!（JS6 = 1 & JS8 = 1）（永远不会出现进路 6 和进路 8 同时建立的情况）

AG!（JS7 = 1 & JS8 = 1）（永远不会出现进路 7 和进路 8 同时建立的情况）

4.3.2 活性 CTL 表达

联锁控制逻辑的活性是指系统的终止性、无活死锁性、保证服务性和响应性等，用于说明“好事情最终会发生”，有以下两种情况：

1）每个信号机都会开放：

EF(S1 = 1)， EF(S2 = 1)， EF(S3 = 1)

EF(S4 = 1)， EF(S5 = 1)， EF(S6 = 1)

2）设置进路，可能会开放相应的信号机，但并不是一定开放。

EF((STATUS = 1)→AF(S1 = 1))

EF((STATUS = 2)→AF(S1 = 1))

EF((STATUS = 3)→AF(S3 = 1))

EF((STATUS = 4)→AF(S5 = 1))

EF((STATUS = 5)→AF(S4 = 1))

EF((STATUS = 6)→AF(S6 = 1))

EF((STATUS = 7)→AF(S2 = 1))

EF((STATUS = 8)→AF(S2 = 1))

这样，用 NuSMV 可以实现对联锁系统的梯形图的建模，生成一个完整的迁移系统，同时，用 CTL 可以很好地表示获得的联锁关系属性，利用 NuSMV 自身提供的功能从模型检测的角度实现安全验证。

4.3.3 模型验证结果

根据图 4 所示的检验结果可以看到：采用 NuSMV 验证工具对上述基于梯形逻辑的车站联锁系统设计模型进行模型监测，全部属性的验证结果均为 true，即用梯形逻辑对计算机联锁控制逻辑的设计符合系统的安全规范。

```
-- specification AG !<DC21 = 1 & S5 = 1> is true
-- specification AG !<DC21 = 0 & S3 = 1> is true
-- specification AG !<DC22 = 1 & S6 = 1> is true
-- specification AG !<DC22 = 0 & S4 = 1> is true
-- specification AG !<JS1 = 1 & JS2 = 1> is true
-- specification AG !<JS1 = 1 & JS3 = 1> is true
-- specification AG !<JS1 = 1 & JS4 = 1> is true
-- specification AG !<JS1 = 1 & JS7 = 1> is true
-- specification AG !<JS2 = 1 & JS3 = 1> is true
-- specification AG !<JS2 = 1 & JS4 = 1> is true
-- specification AG !<JS2 = 1 & JS0 = 1> is true
-- specification AG !<JS3 = 1 & JS4 = 1> is true
-- specification AG !<JS5 = 1 & JS6 = 1> is true
-- specification AG !<JS5 = 1 & JS7 = 1> is true
-- specification AG !<JS5 = 1 & JS0 = 1> is true
-- specification AG !<JS6 = 1 & JS7 = 1> is true
-- specification AG !<JS6 = 1 & JS0 = 1> is true
-- specification AG !<JS7 = 1 & JS0 = 1> is true
-- specification EF S1 = 1 is true
-- specification EF S3 = 1 is true
-- specification EF S5 = 1 is true
-- specification EF S4 = 1 is true
-- specification EF S6 = 1 is true
-- specification EF S2 = 1 is true
-- specification EF <STATUS = 1 -> AF S1=1> is true
-- specification EF <STATUS = 2 -> AF S1=1> is true
-- specification EF <STATUS = 3 -> AF S3=1> is true
-- specification EF <STATUS = 4 -> AF S5=1> is true
-- specification EF <STATUS = 5 -> AF S4=1> is true
-- specification EF <STATUS = 6 -> AF S6=1> is true
-- specification EF <STATUS = 7 -> AF S2=1> is true
-- specification EF <STATUS = 8 -> AF S2=1> is true
```

图 4 NuSMV 验证结果

5 结语

模型检测能避免建立复杂的证明过程，同时能在不满足性质时提供反例，这样可以有效地验证模型，避免设计型故障。采用模型检测方法对用梯形逻辑设计的联锁控制逻辑系统模型进行验证，可以保证系统设计模型的正确性和可靠性，这种自动化验证技术可以大大减少人力资源的浪费。但模型检测在对梯形逻辑表示模型的验证中也存在一些问题有待解决：1）模型检测的主要缺陷就是状态空间爆炸问题，但是解决状态空间爆炸的技术相对于实际系统规模仍有差距；2）模型检测工具要求使用者用特定的系统规范语言刻画梯形逻辑描述的迁移系统，所以使用难度较大。

（下转第 3431 页）

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.