软件学报ISSN 1000-9825, CODEN RUXUEW E-mail: jos@iscas.ac.cn

*Journal of Software*, [doi: 10.13328/j.cnki.jos.000000] http://www.jos.org.cn

©中国科学院软件研究所版权所有. Tel: +86-10-62562563

基于nuXmv的移动机器人永恒探索算法的形式化验证与分析[[1]](#footnote-1)\*

蔡晓伟1,2, 张 民1,2

1(华东师范大学 计算机科学与软件工程学院,上海 普陀 200062)

2(华东师范大学 计算机科学与软件工程学院,上海 普陀 200062)

通讯作者: 张民, E-mail: zhangmin@sei.ecnu.edu.cn

摘 要: 在移动机器人对未知环境空间探索领域，研究发现了一些的机器人探索算法。但是，在研究过程当中，大多学者使用是的使用手动推演的方法，来验证探索算法的正确性和严密性。这种验证方法效率不高，推演过程繁琐，并且很容易出错。为了更加快捷、准确验证机器人探索算法，本文提出了使用nuXmv形式化模型验证工具，对机器人探索算法进行验证和分析。nuXmv工具支持验证LTL公式的验证，可以使用LTL公式描述机器人任意初始状态，并且使用LTL描述机器人探索算法的可满足性。nuXmv对目前提出了完全同步模型（FSYNC，Full-synchronous model）、半同步模型（SSYNC，semi-synchronous model）、异步模型（Asynchronous model）三种探索算法模型都可以建算法模型进行验证，在判定探索算法不满足永恒探索性质时，nuXmv可以给出不满足的状态路径，分析起来十分方便。

关键词: nuXmv; LTL;移动机器人算法;模型验证;

中图法分类号: TP311

中文引用格式: 蔡晓伟，张民. 基于nuXmv的移动机器人永恒探索算法的形式化验证与分析. 第二届全国形式化方法与应用会议（FMAC 2017）.http://www.jos.org.cn/1000-9825/ 0000.htm

英文引用格式: Xiaowei Tsai, Min Zhang. A nuXmv-Based Approach to Formal Verification and Analysis of Mobile Robots Perpetual Exploration Algorithm. Symposium on Dependable Software Engineering, SETTA 2017 (in Chinese).http://www.jos.org.cn/1000-9825/0000.htm

A nuXmv-Based Approach to Formal Verification and Analysis of Mobile Robots Perpetual Exploration Algorithm

CAI Xiaowei1,2, ZHANG Min1,2

1(Shanghai Key Lab of Trustworthy Computing, ECNU, Shanghai 200062, China)

2(MoE International Joint Lab of Trustworthy Software (ECNU), Shanghai 200062, China)

**Abstract**: In the field of mobile robot exploration of unknown environment space, the study found some robot exploration algorithm. However, in the course of the study, most scholars use the method of manual deduction to verify the correctness and rigor of the algorithm. This method of verification is not efficient, the deduction process is cumbersome, and it is prone to error. In order to verify the robot algorithm more quickly and accurately, this paper proposes the verification and analysis of the robot exploration algorithm using the nuXmv formal model validation tool. The nuXmv tool supports verifying the validation of the LTL formula, using the LTL formula to describe any initial state of the robot, and using the LTL to describe the suitability of the robot to explore the algorithm. NuXmv proposed three models of full synchronization model (FSYNC), semi-synchronous model (SSYNC), and asynchronous model (ASYNC). The model can be verified by the algorithm model. The search algorithm does not satisfy the eternal exploration of the nature, nuXmv can give the state of the state is not satisfied, the analysis is very convenient.

**Key words**: nuXmv; LTL; Mobile robots; Verification; Model checking

移动机器人所能完成任务越来越多，任务复杂程度不断增加。许多应用场景设想移动机器人可以在没有任何中央调度的情况下，机器人可以互相协调合作完成任务，这种多机器人系统可以应用于一些特殊环境，比如未知区域的地图构造、环境监测、城市搜索、在危险，人员无法进入的区域进行抢救，监视、未知空间的探索等。在这些应用场景中，自主移动机器人系统的可靠性至关重要，意味着在使用自主移动机器人完成这类任务之前，需要正确和严密验证其可靠性。

本文的研究问题是机器人自主空间探索算法验证方法的改进，机器人自主空间探索算法也可以称为探索协议(exploration protocols)，其中有两种变体：探索终止(exploration with stop)和永恒探索(perpetual exclusive exploration)。在详述探索算法之前，介绍一下机器人的属性。

所有机器人是完全相同，机器人之间无法通过外表进行识别，并且都执行相同的探索算法。并且机器人自身具备以下几个特征：1、无记忆存储：不能存储过去完成的动作。2、无方向传感器：本身无法辨别方向，并且没有偏好移动方向。3、机器人之间无通讯功能：机器人之间不能发送和接收消息。4、但是，机器人有视觉传感器：可以通过视觉传感器获取空间其他机器人的位置信息。

空间模型由原始的连续二维欧几里德空间模型，逐渐演化成为有限位置的离散空间模型。离散空间使用图来描述，图的结点代表空间的位置，图的边表示机器人可以由一个结点到相邻结点的路径。离散空间模型简化了机器人模型，更加关注的是机器人的数量和空间位置之间的关系。但是，在验证某些离散空间(例如：环)的探索算法时，也需要考虑对称性的验证问题。

机器人的移动可以分为三个阶段：观察(Look)，计算(Compute)，移动(Move)。在观察阶段，机器人可以获取图的快照信息，这些快照信息记录着其他机器人在图上的位置。在收集到其他机器人位置信息之后，机器人进入计算阶段，依据收集到的信息计算决定是否移动。移动阶段完成之前计算阶段做出的移动策略。

原始模型中，部分机器人同步执行观察、计算、移动，并且这个三个阶段具有原子性，这种模型下，调度策略有两种：完全同步调度模型(FSYNC，for the fully-synchronous model)和半同步调度模型(SSYNC,semi-synchronous model)。随后，由Flocchini et al提出了异步调度模型(ASYNC，asynchronous, in the sequel)，该模型中，每个机器人观察，计算，移动三个阶段不再具有原子性，每个机器人异步执行移动阶段，也就是说，此时机器人移动可能会使用过时的快照信息，做出移动策略。前面的两种调度策略下，手动验证机器人探索算法，过程十分冗长、复杂，容易出错、遗漏。异步调度模型，由于是异步，存在快照过时的情况，根本无法使用手动推演的方式进行推演验证。

本文的贡献在使用形式化模型验证方法验证永恒探索算法在完全同步调度模型、半同步调度模型、异步调度模型三种模型下的满足性。使用nuXmv描述调度策略模型和探索算法，LTL公式描述机器人未来移动位置的可满足性，验证探索算法是否满足永恒探索要求。当出现不满足情况时，nuXmv可以给出不满足的状态路径，可以清晰分析不满足时的具体细节。近期，Béatrice Bérard的一篇文章中使用DiVinE和ITS工具实现了移动机器人算法的验证，同时Ha Thi Thu Doan使用Maude重写逻辑语言实现移动机器人永恒探索算法的验证。本文提出使用nuXmv的验证方法，验证相同的探索算法时，结果与他们是一致的，nuXmv在时间效率上和Béatrice Bérard所使用的方法相差无几。与前面工作不同的有以下几点：1、nuXmv可以使用LTL公式描述机器人在图上任意初始状态，验证在任意初始状态下探索算法的可满足性。2、使用nuXmv描述探索算法相对比较直观。

文章结构：第一章移动机器人永恒探索算法和调度策略介绍；第二章nuXmv探索算法在不同调度策略下的建模；第三章探索算法的模型检测与结果分析；第四章本文工作与相关工作的对比分析；第五章本文工作总结与未来工作的展望。

# 移动机器人探索算法和调度策略的介绍

本章介绍机器人在图上移动过程和机器人在不同的调度策略下探索算法的差异。为后续的章节nuXmv机器人探索算法建模和模型验证提供预备知识。

## 环境模型

在前面提到，目前环境模型是离散模型，空间位置和机器人移动路径是由图的结点和边表示。可以使用一个集合Rob = {}

## 机器人模型

如图一所示，单个机器人运动过程是观察（Look）、计算（Compute）、移动（Move），然后重新进入观察阶段，重复执行者三个过程。



图一

观察主要是由机器人视觉传感器获取环境快照信息，这里环境信息主要是其他机器人在图上的位置。计算阶段是根据其他机器人的位置信息，匹配机器人设定好的移动算法，根据算法做出移动策略。移动阶段是根据机器人在计算阶段做出的移动策略完成移动动作，此时环境也发生的对应变化。

静态软件缺陷预测可以将程序模块的缺陷倾向性、缺陷密度或缺陷数设置为预测目标.以预测模块的缺陷倾向性为例,其典型研究框架如图1所示.

图1上半部分总结的是软件缺陷预测过程,该过程主要包括两个阶段:模型构建阶段和模型应用阶段.具体来说,模型构建阶段包括3个步骤.

1. 挖掘软件历史仓库,从中抽取出程序模块.程序模块的粒度根据实际应用的场景,可设置为文件、包、类或函数等.随后,将该程序模块标记为有缺陷模块或无缺陷模块,在图1中,我们将有缺陷模块用红色进行标记,无缺陷模块用绿色进行标记;
2. 通过分析软件代码或开发过程设计出与软件缺陷存在相关性的度量元,借助这些度量元对程序模块进行软件度量,并构建出缺陷预测数据集;
3. 对缺陷预测数据集进行必要的数据预处理(例如噪音移除、特征子集选择、数据归一化等)后,借助特定的建模方法构建出缺陷预测模型.大部分建模方法都基于机器学习方法,其常用的模型性能评测指标包括准确率(accuracy)、查准率(precision)、查全率(recall)、*F*-measure或AUC(area under the ROC curve)取值等.

而在模型应用阶段,当面对新程序模块时,在完成对该模块的软件度量后,基于前一阶段构造出的缺陷预测模型和具体度量元取值,可以完成对该模块的分类,即将该模块预测为有缺陷倾向性(defect-proneness,简称FP)模块或无缺陷倾向性(non defect-proneness,简称NFP)模块.



Fig.1 Static software defect prediction research framework using defect-proneness as prediction target

图1 以缺陷倾向性为预测目标的静态软件缺陷预测研究框架

若将预测目标设置为缺陷密度或缺陷数时,其预测流程与图1基本相同,主要的不同点是模型构建阶段中的模块标记(即,需要标记出已有模块内的缺陷密度或缺陷数)和模型预测阶段中的新模块的类型输出(即,预测输出的是新模块内的缺陷密度或缺陷数).

通过分析上述软件缺陷预测过程,我们识别出影响缺陷预测性能的3个重要影响因素(如图1的下半部分所示).

(1) 度量元的设计(见第2节).

挖掘软件历史仓库、设置新颖的与软件缺陷存在强相关性的度量元,是构建高质量缺陷预测模型的关键.本文将已有的度量元分为两类,其中:第一类重点关注的是程序模块的代码规模和内在复杂度;而第二类则重点分析软件开发过程,从分析代码修改特征、开发人员经验、模块间的依赖性以及项目团队组织构架等角度出发来设计度量元.

(2) 缺陷预测模型的构建方法(见第3节).

本文将已有的构建方法分为两类,其中:基于机器学习的方法是目前主流的建模方法,根据预测目标的不同,可以进一步细分为分类方法和回归分析方法;而基于缓存的方法则借助缺陷的局部性原理来尝试识别出缺陷模块.

(3) 缺陷预测数据集的相关问题(见第4节).

本文从两个角度对缺陷预测数据集相关问题进行分析:首先分析了数据集质量对软件缺陷预测的影响,重点对其中的噪音问题、维数灾难问题和类不平衡问题的产生原因及其相应解决方案进行了分析和总结;随后,针对需要预测的目标项目可能是一个全新项目,或这个项目已有的训练数据较少的问题,分析了利用其他项目的数据集来为目标项目构建缺陷预测模型的可行性,并将该问题称为跨项目缺陷预测问题.然后,从实例选择、实例权重设置、特征映射和度量元选择等角度对基于迁移学习的跨项目缺陷预测方法进行了总结.

# 度量元的设计

挖掘软件历史仓库、设置新颖的与软件缺陷存在强相关性的度量元,是构建高质量缺陷预测模型的关键.因此,度量元的设计一直是软件缺陷预测研究中的一个核心问题[4].早期的研究工作主要集中于分析源代码,重点关注基于软件代码(software code)的软件度量.近些年来,更多的研究工作集中于挖掘不同的软件历史存档,重点关注基于软件开发过程(software process)的软件度量.本节将重点从这两个角度出发,对已有研究工作进行系统总结.

## 基于软件代码的度量

在研究早期,大部分研究工作通过分析软件代码来设计度量元.这类度量元重点关注程序模块的代码规模和内在复杂度等属性,其潜在的假设是:代码规模或复杂度越高的程序模块,其内部含有缺陷的可能性越高.

研究人员[5]最早借助代码行数(lines of code,简称LOC)进行度量,例如,Akiyama给出了缺陷数(*D*)与LOC(*L*)的关系式:*D*=4.86+0.018*L*.但该度量元过于简单,难以合理地去度量软件系统的复杂性.随后,研究人员逐渐考虑了Halstead科学度量[6]和McCabe环路复杂度(cyclomatic complexity)[7].其中:

* Halstead科学度量[6]通过统计程序内操作符和操作数的数量来度量代码的阅读难度,其假设是代码的阅读难度越高,其含有缺陷的可能性也越高,涉及到的主要度量元包括程序的长度、容量、难度和工作量等;
* 而McCabe环路复杂度[7]关注的是程序的控制流复杂度,其假设是程序的控制流复杂度越高,其含有缺陷的可能性也越高.在度量时,首先将程序建模为控制流图(control flow graph),其中,节点对应的是语句,边表示从一个语句到另一个语句的控制流.随后,通过公式*v*(*G*)=*e**n*+2计算出控制流图*G*的环路复杂度,其中,*e*表示边的数量,*n*表示节点的数量.最后,可以进一步计算出程序的基本复杂度(essential complexity)和设计复杂度(design complexity).

随着面向对象开发方法的普及,其特有的封装、继承和多态等特性给传统的软件度量提出了挑战.研究人员提出了适用于面向对象程序的度量元,其中最为典型的是Chidamber和Kemerer提出的CK度量元[8].CK度量元综合考虑了面向对象程序中的继承、耦合性和内聚性等特征,给定一个类,其包含的度量元名称及相关描述见表1.

**Table 1** CK metrics

表**1** CK度量元

|  |  |
| --- | --- |
| 名称 | 描述 |
| WMC | 类的加权方法数 |
| DIT | 类在继承树中的深度 |
| NOC | 类在继承树中的孩子节点数 |
| CBO | 与该类存在耦合关系的其他类的数目 |
| RFC | 该类可以调用的外部方法数 |
| LCOM | 类内访问一个或多个属性的方法数 |

Basili等人[9]基于一些中等规模的信息管理系统,首次验证了CK度量元与程序模块内的缺陷存在相关性.随后,Subramanyam和Krishnan[10]基于8个工业界项目,进一步验证了Basili等人的发现.周毓明等人[11]也对基于面向对象程序的度量元与程序模块缺陷间的相关性进行了深入的分析,随后他们[12,13]发现:类规模度量元在分析时存在潜在的混和效应,并会对缺陷预测模型的性能产生影响.因此,他们提出了一种基于线性回归的方法来尝试移除这种混和效应.最后,他们[14,15]分别对Sarkar等人提出的package-modularization度量元[16]和基于程序切片的内聚性度量元[17]与程序模块缺陷间的相关性进行了深入分析.

References:

1. Wang Q, Wu SJ, Li MS. Software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2008,19(7):15651580 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1565.htm
2. Hall T, Beecham S, Bowes D, Gray D, Counsell S. A systematic literature review on fault prediction performance in software engineering. IEEE Trans. on Software Engineering, 2012,38(6):12761304.
3. Yu SS, Zhou SG, Guan JH. Software engineering data mining: A survey. Journal of Frontiers of Computer Science and Technology, 2012,6(1):131 (in Chinese with English abstract).
4. Radjenovic D, Hericko M, Torkar R, Zivkovic A. Software fault prediction metrics: A systematic literature review. Information and Software Technology, 2013,55(8):13971418.
5. Akiyama F. An example of software system debugging. In: Proc. of the IFIP Congress. 1971. 353359.
6. Halstead MH. Elements of Software Science (Operating and Programming Systems Series). New York: Elsevier Science Inc., 1977.
7. McCabe TJ. A complexity measure. IEEE Trans. on Software Engineering, 1976,2(4):308320.
8. Chidamber SR, Kemerer CF. A metrics suite for object oriented design. IEEE Trans. on Software Engineering, 1994,20(6): 476493.
9. Basili VR, Briand LC, Melo WL. A validation of object-oriented design metrics as quality indicators. IEEE Trans. on Software Engineering, 1996,22(10):751761.
10. Subramanyam R, Krishnan MS. Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. IEEE Trans. on Software Engineering, 2003,29(4):297310.
11. Zhou YM, Xu BW, Leung H. On the ability of complexity metrics to predict fault-prone classes in object-oriented systems. Journal of Systems and Software, 2010,83(4):660674.
12. Zhou YM, Leung H, Xu BW. Examining the potentially confounding effect of class size on the associations between object- oriented metrics and change-proneness. IEEE Trans. on Software Engineering, 2009,35(5):607623.
13. Zhou YM, Xu BW, Leung H, Chen L. An in-depth study of the potentially confounding effect of class size in fault prediction. ACM Trans. on Software Engineering and Methodology, 2014,23(1):10:110:51.
14. Zhao YY, Yang YB, Lu HM, Zhou YM, Song QB, Xu BW. An empirical analysis of package-modularization metrics: Implications for software fault-proneness. Information and Software Technology, 2015,57:186203.
15. Yang YB, Zhou YM, Lu HM, Chen L, Chen ZY, Xu BW, Leung H, Zhang ZY. Are slice-based cohesion metrics actually useful in effort-aware post-release fault-proneness prediction? an empirical study. IEEE Trans. on Software Engineering, 2015,41(4): 331357.
16. Sarkar S, Kak AC, Rama GM. Metrics for measuring the quality of modularization of large-scale object-oriented software. IEEE Trans. on Software Engineering, 2008,34(5):700720.
17. Meyers TM, Binkley D. An empirical study of slice-based cohesion and coupling metrics. ACM Trans. on Software Engineering and Methodology, 2007,17(1):2:127.

附中文参考文献:

[1] 王青,伍书剑,李明树.软件缺陷预测技术.软件学报,2008,19(7):15651580. http://www.jos.org.cn/1000-9825/19/1565.htm

[3] 郁抒思,周水庚,关佶红.软件工程数据挖掘研究进展.计算机科学与探索,2012,6(1):131.

1. \* 基金项目: 国家自然科学基金(00000000, 00000000); 南京大学计算机软件新技术国家重点实验室开放课题(KFKT00000000)

   Foundation item: National Natural Science Foundation of China (00000000, 00000000); State Key Laboratory for Novel Software Technology (Nanjing University)开放课题 (KFKT00000000)

   收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos在线出版时间: 0000-00-00

   CNKI在线出版时间: 0000-00-00 [↑](#footnote-ref-1)