1.

2018 届研究生硕士学位论文

分类号：_____ 学校代码：10269

密　级：_____ 学　　号：51151500079

華東師範大學

East China Normal University

硕 士 学 位 论 文

MASTER'S DISSERTATION

论文题目：
自主移动机器人空间永恒探索算法的符号模型检测方法

院　　系：	计算机科学与软件工程学院
专 业 名 称：	软件工程
研 究 方 向：	高可信计算理论与技术
指 导 教 师：	张民 副教授
学位申请人：	蔡晓伟

2017 年 11 月

Dissertation for master degree in 2018 University Code: 10269

Student ID: 51151500079

EAST CHINA NORMAL UNIVERSITY

On Symbolic Model Checking of Mobile Robots Perpetual Exploration Algorithm

Department:	School of Computer Science and Software Engineering
Major:	Software Engineering
Research direction:	Trustworthy Computing Theory and Technique
Supervisor:	Assoc Prof. Zhang Min
Candidate:	Cai Xiaowei

2017.11

华东师范大学学位论文原创性声明

郑重声明：本人呈交的学位论文《自主移动机器人空间永恒探索算法的符号模型检测方法》，是在华东师范大学攻读硕士/博士（请勾选）学位期间，在导师的指导下进行的研究工作及取得的研究成果。除文中已经注明引用的内容外，本论文不包含其他个人已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示谢意。

作者签名:_____ 日期: 年 月 日

华东师范大学学位论文著作权使用声明

《自主移动机器人空间永恒探索算法的符号模型检测方法》系本人在华东师范大学攻读学位期间在导师指导下完成的硕士/博士（请勾选）学位论文，本论文的研究成果归华东师范大学所有。本人同意华东师范大学根据相关规定保留和使用此学位论文，并向主管部门和相关机构如国家图书馆、中信所和“知网”送交学位论文的印刷版和电子版；允许学位论文进入华东师范大学图书馆及数据库被查阅、借阅；同意学校将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于（请勾选）

（）1. 经华东师范大学相关部门审查核定的“内部”或“涉密”学位论文 *，于年月日解密，解密后适用上述授权。

（）2. 不保密，适用上述授权。

导师签名:_____ 本人签名:_____

年 月 日

* “涉密”学位论文应是已经华东师范大学学位评定委员会办公室或保密委员会审定过的学位论文（需附获批的《华东师范大学研究生申请学位论文“涉密”

审批表》方为有效），未经上述部门审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权）。

蔡晓伟 硕士学位论文答辩委员会成员名单

姓名	职称	单位	备注
朱惠彪	教授	华东师范大学	主席
章玥	副教授	华东师范大学	无
毛宏燕	副教授	华东师范大学	无

摘　要

近年来，人类社会不断进步和科学技术飞速发展，机器人逐渐步入了人们的生活。尤其是在电子工业生产、航空航天、汽车生产等领域有着广泛的应用。随着人类活动空间的不断扩大，人类自身的机体能力有限的情况下，寄希望于移动机器人 (autonomous mobile robots) 能代替人自主完成空间探索任务，即机器人根据预设的移动算法，自主根据空间环境中其他机器人的位置，做出对应的移动策略，空间中所有机器人互相协作完成对未知空间的探索。因为机器人在预设定移动算法之后，没有人为进行干预其移动过程，所以这种类型的机器人称为自主移动机器人。

根据具体的物理空间定义移动算法是自主移动机器人完成指定探索任务的核心问题，机器人移动算法决定了机器人的移动行为。移动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。对于机器人移动过程有三种调度模型：完全同步模型 FSYNC(Full-synchronous model)、半同步调度模型 SSYNC(Semi-synchronous model)、异步调度模型 ASYNC(Asynchronous model)。离散空间模型下，已经有了最小移动算法、最大移动算法以及其他的相关研究成果，之前对于这些研究问题的验证过程使用的是手动推演的方式，然而手动推演不仅过程冗长复杂, 推演过程中也容易出现错误. 尤其对于异步调度模型, 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。

形式化方法因其自动性, 严谨性与高效性逐渐被用于各种自主机器人移动算法的验证。使用 DVE 形式化语言描述机器人移动算法，在模型检测工具 DiVinE 和 ITS 中进行验证移动算法是否满足永恒探索。同时也有使用基于重写逻辑的 Muade 形式化建模语言，进行移动算法的建模，验证异步调度调度模型中最小移动算法是否满足永恒探索。

本文以验证自主移动机器人空间永恒探索算法为例, 提出自主移动机器人符号模型检测方法, 该方法不依赖某个具体的初始状态, 且适用于不同的调度模型. 同时, 借助符号模型检测的高效性, 有效避免状态爆炸问题. 利用 nuXmv 符号模型验证工具对机器人探索算法在三种调度模型: 完全同步模型、半同步模型 SSYNC、异步模型进行建模并利用 LTL 公式定义算法的永恒探索性质, 最终实现移动算法的形式化验证. 验证结果表明在假设初始状态未知的条件下依然可验证性质不被满

足并找到反例. 同时, 实验数据表明了符号模型检测对自主移动机器人算法形式化验证的可行性与高效性.

关键词: nuXmv，移动机器人，空间探索，符号模型检测，形式化验证

ABSTRACT

With the development of Internet of Things technology, the role that autonomous mobile robots play in the network is becoming increasingly important. Formal verification of the correctness of autonomous mobile robots has become a new research topic. Most of the exising approaches either suffer state-explosion problem or rely on concrete initial states, which however are usually not undefined and have to be enumerated to make verification complete. In this paper, we propose a symbolic model checking approach to the formal verification of a typical automonous mobile robot system called mobile robot perpetual exploration system. For the symbolicity feature of the model checking, the verifcation does not rely on specific initial states and meanwhile state-explosion problem can be avoided. We use the state-of-the-art symbolic model checker nuXmv to verify the mobile robot perpetual exploration algorithm under three different scheduling modes called full synchronous model (FSYNC), semi-synchronous model (SSYNC) and asynchronous model (ASYNC). Experimental results show that even without providing specific initial states a counterexample can be found in our approach for the perpetual exploration property, which concides with the existing verification result which is obtained by model checking with specific initial states. Meanwhile, the experimental data shows the feasibility and efficiency of symbolic model checking in the formal verification of autonomous mobile robot systems.

Keywords: *nuXmv, LTL, Mobile robots, space exploration, symbolic model checking*

目录

第一章 绪 论 1
1.1 研究背景 1
1.2 国内外研究现状 3
1.3 研究内容和方法 4
1.4 论文结构 5
第二章 nuXmv 介绍 6
2.1 nuXmv 基本数据类型 7
2.2 nuXmv 有限状态机 8
2.2.1 变量声明 8
2.2.2 约束定义 10
2.2.3 模块声明 12
2.2.4 模块实例 13
2.3 LTL 模型检测 15
2.3.1 LTL 的定义 15
2.3.2 LTL 模型验证命令 17
2.4 本章小结 17
第三章 机器人探索算法和调度策略 18
3.1 机器人和探索空间 18
3.1.1 探索空间定义 18
3.1.2 机器人移动三个阶段 19
3.1.3 机器人移动调度模型 19

3.2 环形空间探索算法 . . . 22
3.2.1 机器人视觉快照 . . . 22
3.2.2 机器人移动算法 . . . 24
3.3 探索终止和永恒探索 . . . 27
3.3.1 探索终止 . . . 27
3.3.2 永恒探索 . . . 28
3.4 本章小结 . . . 28
第四章 永恒探索算法建模 . . . 30
4.1 基础模型的构建 . . . 30
4.1.1 机器人建模 . . . 30
4.1.2 移动算法建模 . . . 30
4.1.3 任意初始位置的定义 . . . 30
4.2 调度模型的建模 . . . 30
4.3 本章小结 . . . 30
第五章 永恒探索算法符号模型验证 . . . 31
5.1 机器人初始位置定义 . . . 31
5.2 完全同步调度模型 . . . 31
5.2.1 非碰撞性 . . . 31
5.2.2 非碰撞性 . . . 31
5.2.3 非互换性 . . . 31
5.3 半同步调度模型 . . . 31
5.3.1 非碰撞性 . . . 31
5.3.2 非碰撞性 . . . 31
5.3.3 非互换性 . . . 31
5.4 完全异步调度模型 . . . 31
5.4.1 非碰撞性 . . . 31
5.4.2 非碰撞性 . . . 31
5.4.3 非互换性 . . . 31

5.5 实验结果分析 31
5.6 本章小结 31
第六章 总结和展望 32
6.1 论文总结 32
6.2 工作展望 32
参考文献 33
致谢 33
发表论文和科研情况 34

第一章　绪 论

1.1　研究背景

移动机器人是一种集环境感知、智能决策与规划、行为控制与执行等多种功能于一体的综合系统。它集中了传感器技术、信息处理、电子工程、计算机工程、自动化控制工程以及人工智能等多学科的研究成果，代表机电一体化的最高成就，是目前科学技术发展最活跃的领域之一。随着机器人性能不断地完善，移动机器人的应用范围大为扩展，不仅在工业、农业、医疗、服务等行业中得到广泛的应用，而且在城市安全、国防和空间探测领域等有害与危险工作环境中得到很好的应用。因此，移动机器人技术已经得到世界各国的普遍关注。

自主移动机器人的研究开始于上个世纪六十年代末期。斯坦福大学研究院(SRI) 的查尔斯·罗森 (Charles Rosen) 等人，在 1966 年到 1972 年终研发出了命名为 Shakey 的自主移动机器人。这台机器人被用于人工智能技术的研究，包括复杂环境下机器人的自主逻辑判断、移动路径规划和移动控制。同时，简易的操作式步行机器人也问世，操作式步行机器人步行系统方面的研究是当时的研究热点，解决机器人通过不平整地区的运动不稳定的问题，随后研发和制造拥有多足步行机器人。上个世纪七十年代末，随着计算机嵌入式系统和传感器技术的飞速发展，又迎来了自主移动机器人的研发新高潮。特别是八十年代中期，机器人的研发和设计浪潮席卷全球，当时世界一些著名的科学技术公司着手研制机器人。自从九十年代以来，随着高精度地理环境信息传感器、高速中央处理器、嵌入式控制技术等达到较高水平，以真实物理环境下路径规划技术为标志，对移动机器人开展了更深层次的研究。

许多应用设想在没有中央调度结构的情况下，移动机器人通过自主组织和相互协作，共同完成任务，包括地图的构建、环境检测、城市搜索救援、表面清理、危险区域监控、未知空间的探索等等。在这类的应用中移动算法是保证机器人共同完成任务的关键，意味着移动算法需要进行正式和详尽的验证。在本文中主要关注机器人空间巡逻问题，机器人空间巡逻问题包括巡视一个区域以便检查或保护它，并被广泛的应用于军事、民事领域和一些特殊的系统中。它们可以通过所谓的勘探协议来解决，勘探协议有两种变体，探索终止 (exploration with stop) 和永恒探索 (perpetual exclusive exploration), 分别在文献【f1】和【f1】被提出来。对于探索终止是空间中所有的位置被探索之后，到达某个状态所有机器人永远停止移动。永恒探索是空间中所有机器人都对空间中的每个位置进行反复的访问，永不停止。

移动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，图的结点代表空间中机器人可以到达的位置，图的边表示机器人可以由一个位置结点到达相邻的结点的路径。这样塑造空间模型更加简便直观，更加关注机器人和空间位置结点数量关系对移动算法设定的影响。离散空间模型中，每个位置结点在同一时刻至多只有一个机器人占据。机器人有以下三个特点：1、机器人无记忆存储器，即机器人无法记录历史移动过程;2、无方向传感器，机器人在离散空间中，无法辨别方向，并且没有方向偏好。3、无通讯功能，即机器人之间无法接收或者发送消息；4、拥有视觉传感器，通过视觉传感器机器人可以获取空间中其他机器人的位置信息；5、运算单元，处理移动算法匹配问题，通过计算给机器人下达移动指令，控制机器人的移动；6、移动装置，通过移动装置，机器人可以沿着可移动的路径做出移动动作；7、匿名性，机器人无法通过无表进行区分，即所有机器人都是相同的，且在同一个离散空间系统中，所有机器人执行相同的移动算法。

设计机器人移动算法，并保证机器人设定移动算法之后，能满足探索终止或者永恒探索是目前机器人巡逻问题的一个研究热点。机器人移动算法的设计和验

证，大多是以手动推演的方法为主。手动推演不仅过程冗长复杂，在推演的过程中也容易出现错误或者疏漏，尤其是在某些特定的设置条件下，手动推演方式根本无法进行验证。形式化验证方法因其自动性、高效性、严谨性逐渐被应用于各种移动机器人算法的验证。

nuXmv 是一个新的形式化的符号模型检测工具，不仅集成了最新的 SAT、BDD 模型检测的算法，而且还包含验证性能高效的基于 SMT 的验证技术。使用 nuXmv 工具建模验证时，可以根据实际需求选择验证方式，十分灵活。nuXmv 符号模型检测方法不仅不依赖于某个具体的初始化状态进行验证，而且模块化建模复用性高，大大简化了建模的复杂性。符号模型检测的高效性可以有效避免验证过程中状态爆炸问题。

1.2 国内外研究现状

自主移动机器人领域主要的研究方向有导航和定位、路径规划、多传感器信息融合技术、多机器人系统、机器人仿生技术。影响自主移动机器人研究发展的因素包括传感器技术、运动控制策略、通讯技术、导航和定位系统。进入 21 世纪，随着计算机嵌入式系统和传感器技术的飞速发展，能够供机器人用的高精度传感器不断研发和生产，计算机运算处理能力显著提升，机器人技术已经取得巨大的进展，研究成果硕果累累，然而还远没有满足人类对机器人的需求。

自主移动机器人早期的代表是斯坦福大学研究院 (SRI,Stanford University Research Institute) 研发取名叫 Shakey 的机器人，用于人工智能 (AI,Artificial Intelligence) 技术应用的研究。随着机器人技术的进步，研制出各式各样的移动机器人，按照工作环境来分可以分为室内机器人和室外机器人; 按照机器人移动方式来分有轮式机器人人、步行机器人、履带式机器人等; 按作业空间区分有水下机器人、陆地机器人、无人机、空间机器人等;

本文主要研究是机器人的空间巡查问题，

Lelia Blin 在文献【ring】中提出了环形离散空间中，满足永恒探索的最小

移动算法和最大移动算法，并按照算法的设定，使用手动推演的方式证明了两种算法的正确性。为了提升移动算法验证的准确性、高效性，并能降低验证成本，Béatrice Bérard 等人使用 DVE 形式化语言对机器人移动算法进行建模，通过 DiVinE 和 ITS 验证工具，实现了移动算法的验证。不仅如此，北陆先端科学技术大学院大学 (JAIST) 信息科学学院的 Ha Thi Thu Doan 使用形式化逻辑重写语言 Maude 对

1.3 研究内容和方法

本文的主要研究内容是机器人巡逻问题中自主移动机器人永恒探索算法的形式化方法的改进。机器人移动算法是自主移动机器人协作完成永恒探索任务的核心。空间中所有机器人执行相同移动算法，在没有任何外部控制器的情况下，通过视觉传感器获取的其他机器人的位置信息，与机器人移动算法进行匹配，做出移动决策。机器人不断反复获取空间位置信息、匹配移动算法、做出移动操作，使得每个机器人对离散空间模型中每个位置结点进行反复的访问，称为永恒探索。满足永恒探索的移动算法称为永恒探索算法。本文针对离散空间中机器人在不同的调度模型下移动过程的详细形式化描述和分析。

本文采用符号模型检测的方法实现对机器人移动算法建模，验证移动算法是否是永恒探索算法。nuXmv 是新的符号模型检测工具，能胜任离散空间移动算法的建模验证与分析。对机器人、离散空间、移动算法分别进行模块化建模，模块化代码的可移植性很好，简单清晰，十分方便。完全同步调度模型、半同步调度模型、完全异步调度模型中，模块化代码可以复用，使得复杂的编码过程变得容易，只需针对不同调度模型做出一些对应调整。使用 LTL 公式分别描述永恒探索算法所需满足的非冲撞性、非互换性、非终止性，并可以通过 nuXmv 分别进行验证，在不满足任何一条性质时，nuXmv 可以给出反例状态路径，便于原因分析。为了进一步提升移动算法验证的效率，nuXmv 验证代码通过代码生成器进行生成，只需录入移动算法序列、机器人数等相关信息。

1.4 论文结构

全文一共分为五个章节:

第一章介绍了文章的研究背景，简单描述了自主移动机器人永恒探索算法，并引出了形式化符号模型检测工具 nuXmv，说明使用 nuXmv 工具实现永恒探索算法建模的优点。国内外相关课题的研究现状，以及本文使用 nuXmv 对永恒探索算法的形式化建模验证的方法。

第二章对 nuXmv 工具进行详细介绍，包括符号化建模适用范围，有限状态机的定义，模块化建模，CTL 和 LTL 模型检测命令。通过具体的代码示例描述 nuXmv 的建模和验证过程。

第三章具体介绍探索空间的可能的拓扑结构，详细描述了机器人，探索空间、移动算法的具体内容给出了确切的定义。具体分析自主移动机器人永恒探索算法，包括机器人的移动阶段，机器人移动算法匹配过程，永恒探索的性质。提出三种移动机器人的调度模型，并分别调度模型进行了详细介绍。

第四章提出使用 nuXmv 符号模型检测方法，对自主移动机器人永恒探索算法进行建模验证和分析。以环形拓扑空间的最小移动算法为例，构建最小移动算法在三种不同的调度模型下的模型，并采用 nuXmv 中的基于 BDD 算法和 SMT 的验证指令，进行了形式化的验证，对实验结果也进行详细的分析。

第五章对全文工作做了总结。对实验方法和验证结果进行了分析，总结本文的主要贡献。讨论了 nuXmv 符号模型检测在空间探索协议验证的 BDD 方法和 SMT 方法的选择性，指出了本文工作中的一些不足点，对未来自主机器人领域的研究内容进行了展望。

第二章　nuXmv 介绍

muXmv 是一种新的符号模型验证器，主要使用有限状态系统和无限状态系统的形式化验证和分析。nuXmv 是经典符号模型检测工具 NuSMV 的后续版本，nuXmv 主要从两个方面对 NuSMV 进行拓展：1、对于有限状态系统，采用基于最新的 SAT-based 算法的验证引擎；2、对于无限状态系统，采用基于 SMT 验证技术，通过内部集成 MathSAT5 实现。

nuXmv 在同步系统语言方面进行了扩展，支持 AIGER 规范，在 NUSMV 的基础上新增了整数 (Integer) 和实数 (Reals) 两种数据类型以及解释功能。有限状态系统模型新添加验证算法，基于不变量验证的插值算法、IC3 算法以及 K-liveness 算法。同样，在无限状态系统中引入基于 SMT 验证技术，包括有界模型验证 (Bounded Model Checking)、K-induction、K-liveness、插入值算法 (Interpolation Based)、IC3 算法，以及抽象基础技术 (Abstraction based techniques)。不仅如此，nuXmv 还在功能上进行了扩展，新增 FSM 视图生成、模型的简化和重新编码技术。

nuXmv 继承了 nuXmv 模块化建模特性。每个模块 (Module) 可描述一个状态迁移系统. 模块主要包含三个部分: 变量 (variables), 约束 (constraints) 和规范 (specification). 变量用于定义系统状态, 约束用于定义状态转移关系及模型的一些约束条件。模块化建模不仅使得代码语言简洁清晰，而且可以代码复用，使得建模过程简单。模块化的代码，表达能力更强，逻辑层次明朗。nuXmv 基于已经构建的形式化描述，对嵌入式硬件系统、软件系统相关属性依据数学分析和证明。不仅如此，nuXmv 还可以通过计算逻辑树 (CTL, Computation Tree Logic) 或者线性

时序逻辑 (LTL,inear Temporary Logic) 对系统性质进行描述和验证。因此，nuXmv 验证器在形式化验证领域被广泛的应用，不仅可以为系统提供精准的数学模型，而且可以对系统模型进行形式化的验证和分析。

2.1 nuXmv 基本数据类型

本小节为 nuXmv 支持的数据类型的全部概述，在 nuXmv 中包含布尔类型 (Boolean)、枚举类型 (Enumeration)、字 (Word)、整数 (Integer)、实数 (Reals)、数组 (Array) 和集合 (Set) 集中数据类型。

布尔类型 布尔类型包含符号值真 (TRUE) 和假 (FALSE)。

枚举类型 布尔类型是一个类型的所有值。举例说明，枚举类型的值类似于 $\{stopped, running, waiting, finished\}$，$\{2, 4, -2, 0\}$，$\{FAIL, 1, 3, 7, OK\}$ 等等。枚举类型的值不能有相同的，但枚举的值可以是混合类型的，例如一个枚举同时包含整型数和标识符常量。

字 nuXmv 的字 (Word) 分为无符号字 (unsigned word) 和有符号字 (signed word), 对无符号字和有符号字可以使用按位逻辑和算数操作。字的位宽可以区分，例如 $(unsigned \quad word[3])$ 代表无符号 3 位 (bit) 字，$(signed \quad word[7])$ 代表有符号 7 位 (bit) 字。通用无符号表达式为 $(unsigned \quad word[N])$ 和有符号表达式为 $(signed \quad word[N])$，其中 N 代表该字的位宽 (bit)。

整数 整数有正整数和负整数。首先应该注意的是在某些模型检查引擎和算法中不允许使用整数，其次整型的取值范围 $-2^{32}+1$ 到 $2^{32}-1$。

实数 实数类型的域是有理数。

数组 数组使用索引的上界和下界声明。举例说明，数组 1..8 代表数组中有从 1 到 8 的 8 个整型元素。数组类型与集合类型不兼容，即数组元素不能是集合类型。

集合 集合类型是用于标识一组值的表达式，对于集合只能使用范围常数 (range constant) 和联合运算符 (union)。

2.2 nuXmv 有限状态机

有限状态机 (FSM) 包含状态变量、输入变量和固定变量，在不同状态下这些变量有不同的值。状态迁移 (transition relation) 描绘了某种输入导致系统状态进行转换。公平性条件 (Fairness conditions) 描述了有向状态机的有效路径约束。下面介绍可以被 nuXmv 定义和声明的一些概念。

2.2.1 变量声明

在 nuXmv 中变量分为三种类型：输入 (input) 变量、固定 (frozen) 变量和状态 (state) 变量。变量的声明中需指定变量的的类型。

变量类型定义

变量类型定义的语法如下：

```
type_specifier ::
        simple_type_specifier
    | module_type_specifier

simple_type_specifier ::
          boolean
    | word [ basic_expr ]
    | unsigned word [ basic_expr ]
    | signed word [ basic_expr ]
    | real
    | integer
    | { enumeration_type_body }
    | basic_expr .. basic_expr
    | array basic_expr .. basic_expr
    of simple_type_specifier

enumeration_type_body ::
      enumeration_type_value
    | enumeration_type_body , enumeration_type_value
```

```
20
21 enumeration_type_value ::
22         symbolic_constant
23       | integer_number
```

变量类型定义分为两种：简单类型定义 (simple type specifier) 和模块类型定义 (module type specifier), 模块类型定义在后续的模块实例中进行介绍。简单类型定义包括布尔类型、整数类型、枚举类型、无符号字、有符号类型和数组类型。

状态变量

模型状态就是由一系列状态变量和固定变量的组合，状态变量也称为模块实例 (instances of modules), 用符号表示为：

```
1 var_declaration :: VAR var_list
2
3 var_list :: identifier : type_specifier ;
4           | var_list identifier : type_specifier ;
```

变量声明指定变量的标识符及其类型。变量取值范围是类型取值域。特别是，一个枚举类型的变量只能从其已经定义的枚举值中取值。

输入变量

输入变量 (input variables) 用于标识有限状态机的状态转换。其语法如下：

模型状态就是由一系列状态变量和固定变量的组合，状态变量也称为模块实例 (instances of modules), 用符号表示为：

```
1 ivar_declaration :: IVAR simple_var_list
2
3 simple_var_list ::
4           identifier : simple_type_specifier ;
5       | simple_var_list identifier : simple_type_specifier ;
```

输入变量同状态变脸的不仅声明关键词不同，而且输入变量约定性更强，输入变量不能使模块实例，且比状态变量的使用范围更小。

固定变量

固定变量 (frozen) 是整个状态机中演化过程中保留初始值的变量，固定变量初始值与状态变量以相同的方式进行约束。固定变量的语法同输入变量和状态变量相比只是声明关键词不同:

```
frozenvar_declaration :: FROZENVAR simple_var_list
```

对于一个固定变量 fv 的语义是伴随着赋值并保持其值不变的状态变量，具体含义如下：

```
ASSIGN next(fv) := fv;
```

变量使用举例

下面给出了状态变量、固定变量和输入变量声明的基础用法的例子：

```
    VAR a : boolean;
    FROZENVAR b : 0..1;
    IVAR c : {TRUE, FALSE};
```

变量a 是状态变量，b 是固定变量，c 是输入变量。在下述的例子中：

```
    VAR d : {stopped, running, waiting, finished};
    VAR e : {2, 4, -2, 0};
    VAR f : {1, a, 3, d, q, 4};
```

变量d，e 和f 都是枚举类型。其他的可能的值在其声明中使用类型说明符指定类型。

```
    VAR g : unsigned word[3];
    VAR h : word[3];
    VAR i : signed word[4];
```

变量g 和h 是三个位 (bit) 宽的无符号字，i 是 4 个位宽的有符号字。

```
    VAR j : array -1..1 of boolean;
```

变量g 表示带有索引的布尔元素数组，索引是 -1,0 和 1。

2.2.2 约束定义

INIT 约束 模型的初始状态使用INIT 关键字进行定义，初始状态语法如下：

init_constrain :: INIT simple_expression [;]

INIT 约束中，表达式simple_expression 中不能包含next() 操作。若是模型中定义多个INIT 约束，则可将多个 verb|INIT| 约束表达式并列组合。

INVAR 约束 不变状态可以使用INVAR 关键字定义，INVAR 后面是一个布尔表达式描述数量关系，不变状态语法如下：

invar_constraint :: INVAR simple_expression [;]

类似于INIT 约束，INVAR 约束中，表达式simple_expression 中不能包含next() 操作。

TRANS 约束 模型的转换关系是系统当前状态和系统下一时刻状态的之间的状态迁移关系，使用关键词TRANS 进行定义，当前状态和下一时刻状态之间迁移关系使用布尔表达式来描述。TRANS 的语法如下：

trans_constraint :: TRANS next_expression [;]

语法中表达式next_expression 必须是布尔表达式。若是有多个转换关系，可以将它们并列组合。

ASSIGN 约束 分配 (assignment) 约束的表达形式如下：

```
1 assign_constraint :: ASSIGN assign_list
2
3 assign_list :: assign ;
4               | assign_list assign ;
5
6 assign ::
7     complex_identifier := simple_expr
8     | init ( complex_identifier ) := simple_expr
9     | next ( complex_identifier ) := next_expr
```

在分配定义的左侧，标识符identifier 表示变量的当前值，初始化定义init(identifier) 表示其初始值，状态迁移next(identifier) 表示下一个状态是标识符的值。如果表达式右边是一个非集合表达式，例如整数或者符号常量，意味着左边和右边的相等。另一方面，表达式右边是一个集合表达式，那么意味着左边的取值是在集合元素值范围之内，若是取值不在该范围，模型在验证是，会抛出异常。

语义赋值可以用其他类型的约束表示：

```
ASSIGN a := exp;       is equivalent to INVAR a in exp;
ASSIGN init(a) := exp; is equivalent to INIT a in exp;
ASSIGN next(a) := exp; is equivalent to TRANS next(a) in exp;
```

FAIRNESS 约束 公平性约束仅限于公平执行路径。在 nuXmv 校验模型规范时，模型检测器会使用适用于公平路径的量词。nuXmv 支持两种类型的公平性约束：正义约束 (justice constraints) 和同情约束 (compassion constraints)。正义性约束可以通过关键字JUSTICE 或向后兼容性关键词FAIRNESS 进行定义。同情约束则是由一对程式组合而成(p,q)，若程式p 在无限状态路径中公平取值为真，那么程式q 也会在无限状态路径中取值为真，在 nuXmv 中同情约束使用关键词COMPASSION 进行定义。

公平性的语法定义如下：

```
fairness_constraint ::
    FAIRNESS simple_expr [;]
    | JUSTICE simple_expr [;]
    | COMPASSION ( simple_expr , simple_expr ) [;]
```

当且仅当状态路径满足上述定义的所有公平性的约束才称为公平状态路径。

2.2.3 模块声明

nuXmv 模块声明是一个封装声明，是包含声明、约束和规范的集合。模块声明是一个新的标识符范畴。模块一旦被定义，就可以在有必要时被重复使用。模块定义中可以包含其他的模块实例。nuXmv 中模块的语法如下：

```
module :: MODULE identifier [( module_parameters )] [module_body]

module_parameters ::
        | module_parameters , identifier

module_body ::
          module_element
        | module_body module_element

module_body ::
          module_element
        | module_body module_element
module_element ::
```

```
14          var_declaration
15        | ivar_declaration
16        | frozenvar_declaration
17        | define_declaration
18        | constants_declaration
19        | assign_constraint
20        | trans_constraint
21        | init_constraint
22        | invar_constraint
23        | fairness_constraint
24        | ctl_specification
25        | invar_specification
26        | ltl_specification
27        | pslspec_specification
28        | compute_specification
29        | isa_declaration
30        | pred_declaration
31        | mirror_declaration
```

模块定义中紧跟在关键词MODULE 后的标识符，是该模块的名称。模块名称在 nuXmv 程序中拥有一个单独的命名空间，因此可能与变量或者定义的名称发生冲突。

2.2.4 模块实例

nuXmv 可以描述有限状态机，包括完全同步和完全异步状态机。可以描述一个同步的 Mealy 机，也可以描述一个不确定性过程的异步网络模型。nuXmv 的输入型语言提供模块化层次描述、可重用组件的定义。因为其设计目的是描述有限状态机，其数据类型都是有限的，例如布尔类型、标量和固定数组。也可以构建静态数据类型。

nuXmv 输入数据类型主要是用来描述有限状态机中的状态迁移。状态迁移描述了有限状态机的状态演变。通常，命题表达式都可以被用来描述状态迁移。这虽然提供了很大的灵活性，同时也隐含某种危险，例如，逻辑矛盾可能导致状态死锁 (deadlock)。虽然可以使用模型检查过程的方式来检测状态死锁问题，但是最好是通过限制描述的方式来避免这个问题。nuXmv 通过并行分配的语法 (parallel-assignment syntax) 支持这点。

同步系统 -单进程实例 下面给出一个同步系统的单进程实例：

```
MODULE main
VAR
    request : boolean;
    state : {ready, busy};
ASSIGN
    init(state) := ready;
    next(state) := case
                        state = ready & request = TRUE : busy;
                        TRUE                : {ready, busy};
                   esac;
```

有限状态机的状态空间是由状态变量的声明决定的，在上述例子中的request和state。变量request 是布尔类型，变量state 是一个标量变量，可以接受符号值ready 或者busy。接下来使用分配约束给state 初始化值为ready。而变量request初始值是完全未指定的，即其初始值可能是TRUE 或者FALSE。状态机的状态迁移使用下一个状态 (next) 来进行定义，给出当前状态和下一个状态之间的关系，case 类似 C 语言中的条件判断语句 $if..elseif..else$ 一样，当满足条件 $state = ready$ 和 $request = TRUE$ 时，下一个状态state 变量的取值为busy，其他情况下则取值为ready 或者busy。

异步系统 -反相门环 nuXmv 也允许异步系统建模。在模型内部定义并行的进程集合，进程的操作是交错的。异步系统模型常常用来描述通讯协议、异步电路以及其他异步系统。下面是反相门环 (Inverter Ring) 异步系统建模的例子：

```
MODULE inverter(input)
    VAR
        output : boolean;
    ASSIGN
        init(output) := FALSE;
        next(output) := !input;

MODULE main
    VAR
        gate1 : process inverter(gate3.output);
        gate2 : process inverter(gate1.output);
        gate3 : process inverter(gate2.output);
```

在上述程序中定义了由三个反相门构建的环，反相门模块使用关键字process实例化。每个反相门实例类似计算机系统中的进程 (process), 给每个反相门模块实例随机分配时间片，当反相门获取时间片时，执行其中的赋值语句。由于是随机

分配时间片给每个反相门，可能会出现某个或者几个方向门实例难以分配到执行时间片。为了保证每个进程公平的获得执行时间片，可以在每个模块定义中添加公平性约束：

```
FAIRNESS
  running
```

2.3 LTL 模型检测

nuXmv 中可以通过时序逻辑对有限状态机进行建模与验证，如计算树逻辑 (CTL，Computation Tree Logic)、线性时态逻辑 (LTL，Linear Temporal Logic)、属性说明语言 PSL(Property Specification Language)。还可以使用实时计算树逻辑 (real-time CTL) 来分析有限状态机的数量特征。使用时序逻辑 CTL 或 LTL 描述预验证性质，可以验证有限状态机对预验证性质的满足性。若有限状态机不满足预验证性质，nuXmv 则返回 false 并构建和打印出一个不满足预验证性质的反例。反例内容是有限状态机不满足预验证性质的反例状态路径。使用时序逻辑描述预验证性质，不仅简化了建模代码，而且可移植性强。由于本文实验中主要使用的是 LTL 描述机器人永恒探索算法的性质，所以在本章节中只介绍 LTL 的定义和 LTL 模型检测命令。

2.3.1 LTL 的定义

nuXmv 中通过LTLSPEC 可以定义一条 LTL 表达式。当系统模型中有多个性质需要验证其满足性时，可以对应声明多个 LTL 表达式进行性质描述。具体的 LTL 表达式语法如下：

```
ltl_specification  ::  LTLSPEC ltl_expr [;]
                       LTLSPEC NAME name := ltl_expr [;]
```

从上述语法中 LTL 表达式有两种声明方式，LTLSPEC 关键词声明匿名表达式，而LTLSPEC NAME 关键词声明是名称为 name 的 LTL 表达式。

通过一个例子对 nuXmv 中 LTL 表达式进行详细的介绍，下面代码定义了一个异步信号系统，包含一个主模块main 和用户模块user。

```
MODULE main
    VAR
        semaphore : boolean;
        proc1 : process user(semaphore);
        proc2 : process user(semaphore);
    ASSIGN
        init(semaphore) := FALSE;
    LTLSPEC G ! (proc1.state = critical & proc2.state = critical)
    LTLSPEC G (proc1.state = entering -> F proc1.state = critical)

MODULE user(semaphore)
    VAR
        state : {idle, entering, critical, exiting};
    ASSIGN
        init(state) := idle;
        next(state) :=
            case
                state = idle : {idle, entering};
                state = entering & !semaphore : critical;
                state = critical : {critical, exiting};
                state = exiting : idle;
                TRUE : state;
            esac;
        next(semaphore) :=
            case
                state = entering : TRUE;
                state = exiting : FALSE;
                TRUE : semaphore;
            esac;
    FAIRNESS
        running
```

用户模块带有一个外部传入参数 semaphore，在模块内部声明了一个枚举类型的状态变量state，并给变量分配初始值idle。next(state) 表示下一个状态中 state 的值：当前状态下 state 等于 idle 时，下一个状态中 state 的取值是 idle 或者 entering；当前状态下满足 state 等于 entering 且 semaphore 为 FALSE 时，下一个状态中 state 的取值是 critical；当前状态下满足 state 等于 critical 时，下一个状态中 state 的取值是 critical 或者 exiting；当前状态下满足 state 等于 exiting 时，下一个状态中 state 的取值是 idle；TURE 表示不满足上述几种情况下，下一个状态中 state 等于当前状态下 state 的值。同样，next(semaphore) 表示下一个状态下semaphore 的值：当前状态下 state 等于 entering 时，下一个状态中 state 的取值是 TRUE；当前状态下 state 等于 exiting 时，下一个状态中 state 的取值是 FALSE；TURE 表示在不满足上述情况下，下一个状态中 semaphore 等于当前状态的 semaphore。并且给用户模块

设定了公平性约束 FAIRNESS。在主模块中，声明了布尔类型的变量 semaphore，并给其初始值 FALSE。在主模块中实例化两个用户模块 proc1 和 proc2，并分别使用关键词 process 修饰，表明两个用户实例化模块为异步。还声明了两个 LTL 表达式，LTL 表达式LTLSPEC G ! (proc1.state = critical & proc2.state = critical) 中 G 表示将来所有时刻都满足，而整个 LTL 表达式含义是当用户模块 proc1 的 state 变量取值为 critical 时，用户模块 proc2 的变量 state 不能同时取值为 critical。同样 LTL 表达式LTLSPEC G (proc1.state = entering -> F proc1.state = critical) 说明若只要用户模块 proc1 的 state 变量取值为 entering 时，在此之后的某个状态下必定取值为 critical。

2.3.2 LTL 模型验证命令

nuXmv 在其诞生的过程中，不仅融合很多优秀的验证算法，而且还集成了 SMT 解析器 MathSAT5 和最新的 SAT 解析器。所以，nuXmv 对于不同的时序逻辑都设计了丰富的验证执行命令。基于 BDD 的验证方法在有限状态机满足预验证性质时，验证效率较高，而基于 SMT 验证方法在有限状态机不满足预验证性质时，很容易找出不满足的反例。不同验证方式适用于不同的验证场景，在此主要介绍一下基于 BDD 的增量 COI 算法验证命令和基于 SMT 的增量 COI 算法验证命令。

基于 BDD 的增量 COI 算法验证命令 BDD 是指二元判决图 (Binary Decision Diagram), 对于布尔函数运算的效率很高。

```
1 check_ltlspec_inc_coi_bdd [-h] | [-n number | -p "ltl-expr [IN context]"| -P "name"] [-l]
```

基于 SMT 的增量 COI 算法验证命令

```
1 msat_check_ltlspec_inc_coi [-h] | [-n number | -p "ltl-expr [IN context]" | -P "name"] [-l] [-u] [-i] [-k bound]
```

2.4 本章小结

本章介绍了符号模型验证器 nuXmv 的基础数据类型，通过基础数据类型和模块定义可以实现有限状态机的建模。nuXmv 支持多种时序逻辑语言，本章节主要

是讲解通过其中的 LTL 定义有限状态机中预验证性质的可满足性验证。并以异步信号系统为例，详细的介绍了模型的构建以及模块中状态迁移，重点说明 LTL 在模块中声明。此外，还阐述了 nuXmv 支持的各种 LTL 的验证方法，对于验证性质的正确性,BDD 方法的效率相对较高, 而对于不被满足的性质, 基于 SMT 验证方法可以更快的找到不满足的反例。

第三章　机器人探索算法和调度策略

3.1　机器人和探索空间

3.1.1　探索空间定义

离散模型中，探索空间一般抽象为一个无向连通图，图的每个结点代表空间上机器人可达的位置，边代表机器人可以通过的路径，机器人沿着该路径到达相邻的空间位置。空间中每个位置结点在同一个时间至多只有一个机器人。

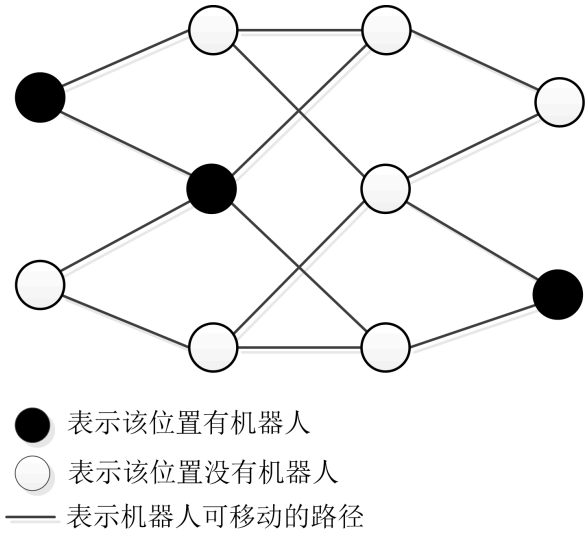

图 3.1: 无向连通图表示探索空间

如图3.1给出了一个简单的离散探索空间的无向连通图表示，黑色结点表示该空间位置在此刻有一个机器人，白色结点说明该空间位置上没有机器人，图中有三个机器人，分别位于三个黑色的空间结点上。类似于计算机网络拓扑结构，探索空间结构也有总线拓扑结构、星型拓扑结构、环形拓扑结构、树形拓扑结构等。

3.1.2 机器人移动三个阶段

离散空间上每个机器人的移动分为三个阶段，分别是观察 (look)、计算 (compute) 和移动 (move). 在观察阶段，机器人通过自身的视觉传感器，获取空间环境中其他机器人的位置快照信息。然后进入计算阶段，计算阶段根据观察阶段获取的位置快照信息，匹配自身预先设置的移动算法，计算得出下一步的移动策略。移动策略包括机器人是否移动，若是移动，则是沿着那条路径进行。移动阶段，机器人的动力装置按照计算阶段的所得移动决策作出对应的移动。

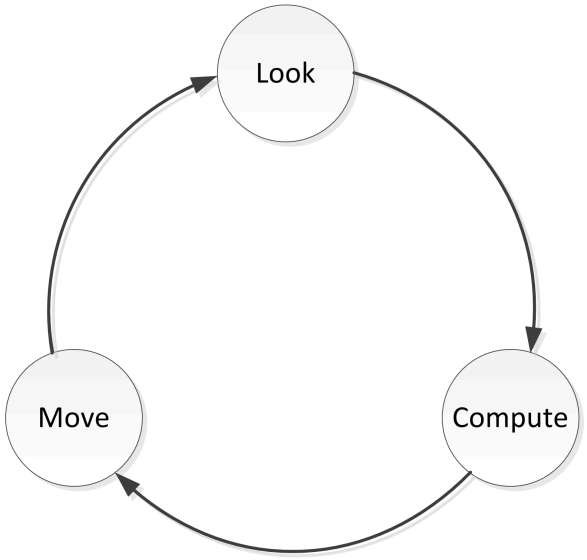

图 3.2: 机器人移动三阶段

如图3.2描述了机器人移动的三个阶段，这三个阶段按照观察阶段、计算阶段、移动阶段，完成移动阶段之后又进入观察阶段，这样不断重复三个移动阶段。

3.1.3 机器人移动调度模型

原有模式中，Suzuki 假设空间中机器人个数大于 0，机器人的移动三个阶段具有原子性，且机器人之间的运动过程是同步的，提出了移动调度策略的两种变体，分别是完全同步调度模型 FSYNC(fully-synchronous model) 和半同步调度模型 SSYNC(semi-synchronous model)。后来由 Flocchini 等人提出了异步调度模型 ASYNC (asynchronous model), 该异步调度模型中，机器人观察, 计算, 移动三个移

动阶段不再具备原子性，这样可能会出现机器人使用过时的快照信息做出移动决策。

下面使用数学和算法知识来描述完全同步调度模型、半同步调度模型、异步调度模型具体的内容。

假设在一个离散空间中存在 $k\,(k \in N^{+})$ 个机器人，使用集合 $Rob = \{r_1, r_2, r_3, ..., r_k\}$ 表示。空间位置结点可以按照一定顺序，对其进行编号，编号是空间位置结点的唯一标识，使用集合 $Pos = \{n \in N^{+} | 0, 1, 2, ..., n-1\}$ 表示空间所有空间位置结点，每个元素对应唯一一个空间位置结点。机器人是在空间位置结点上的，所以机器人与空间位置结点的映射关系使用 $p : \{Rob \to Pos\}$ 表示，机器人 r 在某个时间所在图中的位置结点编号就为 $p\,(r \in Pos)$。对于 Rob 中的任意两个机器人 $r_i, r_j\,(i \neq j)$, 在任意一个时刻满足位置结点不相同 $p\,(r_i) \neq p\,(r_j)$, 即空间中任意时刻一个位子结点上至多只有一个机器人。

前面使用数学中的集合和映射定义了探索的离散空间、机器人已经机器人在空间结点上的函数关系，在此基础上，下面将介绍机器人移动的三种调度模型。完全同步调度模型是半同步调度模型的一种特殊情况，所以先介绍半同步调度模型，在介绍完全同步调度模型。完全异步调度模型完全与前面两种调度模型由较大区别，所以放在最后介绍。

将机器人移动的一个完整的观察、计算、移动称为完整移动阶段，在半同步调度模型中，Rob 集合中只有选中的机器人才进行完整移动阶段。而半同步调度模型所有机器人都是同步而且观察、计算、移动都是具备原子性，所以可以每次选中的机器人是一个非空集合 $Sched \subseteq Rob$, 在一个完整移动阶段开始之前，只有选入集合 Sched 的机器人才会执行完整移动阶段，即 $\forall r \in Sched$ 执行观察、计算、移动，而 $\forall r \notin Sched$ 不执行。等到下一个完整移动阶段之前，又会从 Rob 随机选择一个机器人 $Sched \subseteq Rob$，重复上述过程。这就是半同步调度模型机器人调度的性质。下面使用简易算法过程，详细描述一下半同步调度模型中机器人执行完整移动阶段的过程。

```
1 SSYNC-SCHEDULE(Rob)
```

```
  while
    choose Sched from Rob
    synchronous {
       foreach r in Sched{
          r.look
          r.compute
          r.move
       }
    }
```

半同步调度模型 SSYNC-SCHEDULE 的传入参数是机器人集合 Rob, 首先从 Rob 集合中选择子集合 Sched 且 $Sched \neq \emptyset$。关键字synchronous 表示同步块中所有机器人同步完成移动阶段。在 Sched 集合中的每个机器人开始同步执行观察、计算、移动。完成之后，又重新开始随机选择子集合 Sched，重复不断执行上述过程。

完全同步调度模型是半同步调度模型中一种很特殊的情况，每次随机选择的集合 $Sched = Rob$, 即每次完整移动阶段之前在集合 Rob 中所有的机器人都被选中。使用算法过程描述如下：

```
FSYNC-SCHEDULE(Rob)
  while
    synchronous {
       foreach r in Sched{
          r.look
          r.compute
          r.move
       }
    }
```

同半同步调度模型相对较而言，完全同步调度模型只是在每次选择集合 $Sched$ 有所不同，其他过成完全相同。

而完全异步调度模型中，所有的机器人观察、计算、移动都是异步，没有原子性。类似于计算机系统的多线程，每个机器人的移动都是并行且互相之间没有同步约束，当一个机器人在观察时，其他机器人可能在执行计算或者移动。每个机器人执行移动的快慢完全是随机的，所以肯能会出现机器人在观察阶段通过视觉传感器获得快照是过时的。

```
ASYNC-SCHEDULE(Rob)
    asynchronous {
       foreach r in Rob{
```

```
4            while{
5               r.look
6               r.compute
7               r.move
8            }
9        }
10   }
```

上述完全异步调度模型算法中，关键字synchronous 表示 Rob 中所有的机器人都异步执行移动，对于每个机器人而言，都是在按照顺序不断重复执行观察、计算和移动过程，即整个过程中机器人都是并行执行完整移动阶段。

3.2 环形空间探索算法

探索空间结构有总线拓扑结构、星型拓扑结构、环形拓扑结构、树形拓扑结构, 不同的探索空间结构有各自的特点，本文以环形拓扑结构空间为例. 首先介绍环形拓扑结构环上的机器人视觉快照、匹配移动算法获取移动决策、移动决策的执行. 在此基础上, 将介绍永恒探索移动算法的相关概念.

3.2.1 机器人视觉快照

图3.3给出一个简单的环形拓扑结构探索空间的例子。根据环形空间的自身特点，沿着环的顺时针方向，从0 开始递增进行编号。所有位置编号组成的集合为Pos，图中黑色结点表示该位置结点上有机器人，白色结点表示该位置结点上没有机器人。下面给出某个时刻某个结点上机器人的数量的定义。

定义 1 (机器人所在位置编号) 对于机器人r，其所在探索空间中位置的编号使用符号 $c_{(r)}$,$c_{(r)} \in Pos$。

定义 2 (结点机器人个数) 结点机器人数 d_j，j 表示结点的编号 $j \in Pos$，当结点j 上有 $k\,(k>0)$ 机器人表示为 $d_j = k$, 本文中同一时刻每个结点至多有一个机器人，在结点上存在机器人时 $d_j = 1$，当结点j 没有机器人时，表示为 $d_j = 0$。

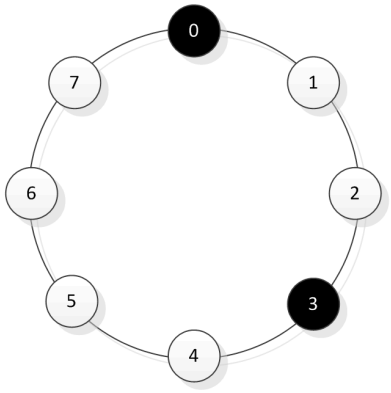

图 3.3: 环形拓扑结构探索空间

在环形拓扑结构的离散空间中，空间中的每个机器人可以顺时针观察，也可以逆时针观察。那么机器人每次观察的位置信息快照就有顺时针位置信息快照和逆时针位置信息快照两种，为了方便描述，给出某个位置上机器人获取位置信息快照的定义，如下：

定义 3 (机器人快照) 离散空间上位置结点 $p \in Pos$，j 上的机器人通过视觉传感器获取的位置快照为 δ_p^F，其中 $F \in \{+,-\}$,+ 表示顺时针，-表示逆时针。

在拥有n 个位置结点的环形拓扑结构的探索空间上，任意位置结点j 上机器人的顺时针和逆时针快照如下：

顺时针序列定义： $\delta_p^+ = \langle d_j, d_{j+1}, ..., d_{j+n-1} \rangle$.

逆时针序列定义： $\delta_p^- = \langle d_j, d_{j-1}, ..., d_{j-n+1} \rangle$.

如图3.3中以位置编号为 0 的结点为例，其结点上机器人的顺时针和逆时针位置信息快照如下：

顺时针序列1： $\delta_0^+ = \langle 1,0,0,1,0,0,0,0 \rangle$.

逆时针序列1： $\delta_0^- = \langle 1,0,0,0,0,1,0,0 \rangle$.

虽然上述位置快照信息描述比较简洁和直观，但是当空间结点数 n 较大时，位置快照信息就过长，不利于描述。后来由 lelia.blin 在其文献 [Ring] 中提出了一种新的位置快照F-R 表达方式，F-R 表达方式中使用 F_m 表示连续的m 个空间位

置结点上没有机器人，R_n 表示连续的n 个空间位置结点上有机器人。那么顺时针序列 1 和逆时针序列 1 转化为 F-R 的表达方式为：

顺时针F-R序列1： $\delta_0^+ = \langle R_1, F_2, R_1, F_4 \rangle$.

逆时针F-R序列1： $\delta_0^- = \langle R_1, F_4, R_1, F_2 \rangle$.

定义 4 (机器人快照 F-R 表达式)k($k \in N^+$) 个机器人在n ($n \in N^+$) 个位置结点环形空间中，且 $k < n$, 使用符号R 和F 所组成的符号序列表示机器人快照。带下标的 R_i 表示连续i 个空间位置结点，每个位置结点被一个机器人占据。带下标的 F_j 表示连续j 个空间位置结点，每个位置结点上都没有机器人。

机器人快照F-R 表达式不仅可以解决环形空间结点数 n 较大，不易描述的问题，而且可以通过下角标使用未知变量，表达连续没有机器人位置结点数和有机器人位置结点数，增强了表达式的灵活性、可读性和表达能力。

3.2.2 机器人移动算法

机器人移动算法是保证自主机器人协同完成设的任务的核心，在同一个探索空间中的机器人都预先设置完全相同的移动算法。机器人的可能的移动策略取决于其所在探索图形状，本小节主要介绍环形探索图中的机器人移动算法。

环形探索图中，对于某个位置结点g 上的机器人来说，若其视觉快照 $\delta_g^+ = \delta_g^-$，称为对称 (symmetry)。由于对称情况的存在，所以机器人的移动存在四种方式：不移动 ($Idle$)、前进 ($Front$)、后退 ($Back$)、未确定 ($Doubt$)。未确定的情况是位置结点上的机器人顺时针视觉快照和逆时针视觉快照相同，那么该结点上机器人可能前进或者后退。未确定表示此时机器人随机选择前进或者后退。

本文中的机器人移动算法，是机器人快照 F-R 表达式和对应四种移动方式组合而成。下面使用机器人最小移动算法为例，详细介绍机器人移动算法。最移动算法是指环形探索空间中，空间位置结点数 $n \geq 10$, 图上的机器人 $k = 3$，且n 和k 数量关系上互质。最移动算法分为两个阶段：稳定阶段 ($Legitimate \quad phase$) 和收

敛阶段 ($Convergence \quad phase$)。

表 3.1: 最小移动算法稳定阶段

$RL1$	$\delta^F_{c(r)} = \langle R_2, F_2, R_1, F_{n-5} \rangle .$	$\rightarrow$	$r.Back$
$RL2$	$\delta^F_{c(r)} = \langle R_1, F_1, R_1, F_{n-6}, R_1, F_2 \rangle .$	$\rightarrow$	$r.Front$
$RL3$	$\delta^F_{c(r)} = \langle R_1, F_3, R_2, F_{n-6} \rangle .$	$\rightarrow$	$r.Front$

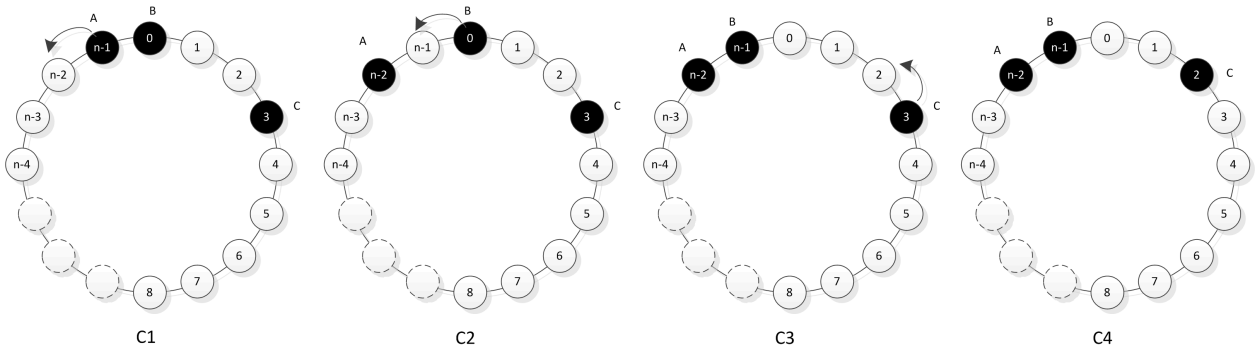

图 3.4: 最小移动算法的稳定阶段

在图3.4中，给出了三个机器人按照预设的稳定阶段算法运动过程的例子，C1 中机器人 A 的顺时针位置信息快照 $delta^+_{c(A)}$ 匹配移动算法 $RL1$ 做出后退的移动决策，机器人 A 从位置结点 $n-1$ 移动到 $n-2$, 其他机器人 B 和 C 此时没有匹配任何移动算法，保持静止。C2 中机器人 B 的逆时针位置快照 $delta^-_{c(B)}$ 匹配移动算法 $RL2$，做出前进的移动决策，机器人 A 从位置结点 0 移动到 $n-1$。同样机器人 C 的逆时针位置快照信息，匹配移动 $RL3$ 做出前进的移动决策。图中 C1 和 C4 进行对比，图中三个机器人的位置都向顺时针移动了一位，C4 中 A 获取的位置信息 F-R 表达式同 C1 中的 A 获取的位置信息 F-R 表达式相同。意味着整个系统会按照上述 C1 到 C4 过程重复执行，而且每次执行之后，所有机器人都会向逆时针方向移动一位。这说明系统进入了稳定阶段，稳定阶段的系统状态称为稳定状态。系统进入稳定状态之后，就会一直停留在稳定阶段中。介绍了最小移动算法的稳定阶段之后，下面介绍其收敛阶段。

表 3.2: 最小移动算法收敛阶段

$RC1$	$4 \leq x \leq z \wedge \delta^F_{c(r)} = \langle R_1, F_x, R_2, F_z \rangle$.	$\rightarrow$	$r.Front$
$RC2$	$x \neq y, x > 0 \wedge \delta^F_{c(r)} = \langle R_1, F_x, R_1, F_y, R_1, F_x \rangle$.	$\rightarrow$	$r.Doubt$
$RC3$	$0 < x < z < y \wedge (x, y) \neq (1, 2) \wedge \delta^F_{c(r)} = \langle R_1, F_3, R_2, F_{n-6} \rangle$.	$\rightarrow$	$r.Front$
$RC4$	$\delta^F_{c(r)} = \langle R_3, F_{n-3} \rangle$.	$\rightarrow$	$r.Back$
$RC5$	$\delta^F_{c(r)} = \langle R_1, F_1, R_2, F_{n-4} \rangle$.	$\rightarrow$	$r.Back$

在表**??**中，是最小算法的收敛算法，系统状态除了稳定状态之外的都称为收敛状态，系统任意收敛状态都可以通过收敛算法到达稳定状态。下面将以最小移动算法为例，具体介绍系统如何从任意的一个状态到达稳定状态。

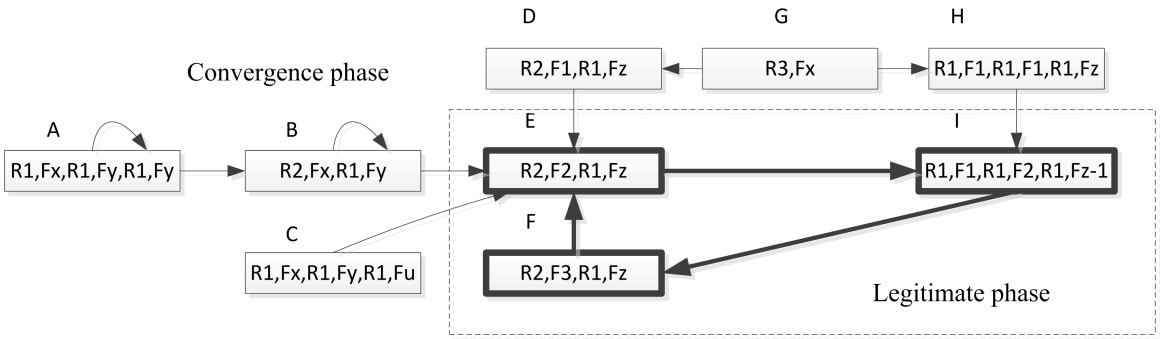

图 3.5: 最小机器人移动算法状态转移

图3.5，描绘了完全同步调度模型和半同步调度模型中，最小移动算法中的稳定阶段和收敛阶段的所有状态，移动机器人所在空间位置结点上的视觉快照，匹配移动算法中状态，满足则按照对应的移动策略做出移动动作。每当机器人做出移动之后，空间中位置信息就会发生变化，系统的状态发生改变。虚线方框中的表示稳定阶段的三种状态，构成了有向强连通图。进入稳定阶段的状态，就会在三种状态之间按照箭头指向顺序，不断重复转换，再也不能进入除此三种状态以外的状态了。对于虚线以外的状态称为收敛状态，收敛状态可以通过收敛算法逐渐到达稳定状态。

3.3 探索终止和永恒探索

由 Flocchini 在文献 [1] 提出了探索终止 (exploration with stop) 问题，所谓探索终止是指空间中每个位置结点都会被至少一个机器人访问一次，在空间上每个结点都被访问之后，所有机器人都停止不动。Flocchini 证明了环形空间，位置结点数 n 和机器人数 k ，若 n 对 k 取模为 0，那么没有移动算法满足探索终止性质。同时，文献中给出一组确定性的移动算法, 在机器人数 $k \geq 17$, 机器人数 k 与结点 n 互质时，满足探索终止性。永恒探索 (perpetual exclusive exploration) 是指空间中每个机器人对空间中的每个结点重复的访问，并且该过程是不停止的。

探索终止和永恒探索是都是自主机器人对空间位置结点的巡查模式，这两种巡查模式都是确保机器人协作完成预设探索空间任务的方式。探索终止和永恒探索只有微小区别，在本质上是所有空间位置结点都被机器人访问。本小节介绍探索终止和永恒探索具体内容。

3.3.1 探索终止

对于环形探索空间来讲，无论空间中所有机器人在什么初始位置，给出一个移动算法使得机器人组在有限时间内满足探索终止。这可以归纳为两条性质，如下：

探索性 (Exploration): 环上每个结点至少被一个机器人访问。

终止性 (Termination): 在满足探索性前提下，最终到达某一状态使得所有的机器人都保持静止，不再移动！

对于第二条性质而言，要求机器人” 记住” 环形探索图中有多少位置结点已经被访问了，即这些没有记忆存储的机器人通过它们当前获取的视觉快照，区分探索过程中不同阶段。这两种属性也可以使用 LTL 公式来表示，如下：

探索性 (Exploration): $\bigwedge_{j=0}^{n-1} \Diamond (d_j > 0)$。

终止性 (Termination): $\bigwedge_{j=0}^{n-1} \Diamond (d_j > 0)$。

3.3.2 永恒探索

对于一个环形探索空间中，机器人任意初始状态，每个机器人在预设的移动算法下，都可以对环形探索空间中的任意一个位置结点无限反复的进行访问。对于同于的位置结点而言，在同一时刻至多只能有一个机器人占据，称为非冲撞性 (No collision)。对于相邻的两个机器人，不能同时通过相邻边互换空间位置，即相邻的两个机器人不能在相邻的边上发生碰撞，该性质称为非互换性 (No switch)。环形图中每个机器人都必须对探索空间中任何一个位置结点进行反复的访问，且永不停止，称为非终止性 (Live)。上述三条性质就是永恒探索内在意义，满足上述三条性质的算法称为永恒探索算法。下面使用 LTL 公式分别描述这三条性质：

非冲撞性 (No collision): $\bigwedge_{i=1}^{k}\bigwedge_{j=1}^{k}\square\left(i\neq j\rightarrow c_{r_i}\neq c_{r_j}\right)$。

非互换性 (No switch): $\bigwedge_{h=0}^{n-1}\bigwedge_{i=1}^{k}\bigwedge_{j=1}^{k}\square\left(c\left(r_i\right)=h\wedge c\left(r_j\right)=(h+1)\,mod\,n\rightarrow\left(\urcorner\left(c\left(r_i\right)=(h+1)\,mod\,n\right)\wedge c\left(r_j\right)=h\right)\right.$。

非终止性 (Live): $\bigwedge_{h=0}^{n-1}\bigwedge_{i=1}^{k}\square\diamond\left(c\left(r_i\right)=h\right)$。

上述三条 LTL 公式描述性质中，n 表示探索空间的位置结点数，k 表示探索空间中的机器人数。在异步调度模型中，上述三条性质都必须在公平性为前提，即任何一个机器人都可以被无限次的调度，才能满足永恒探索。公平性的 LTL 公式描述如下：

公平性 (Fairness): $\bigwedge_{i=1}^{k}\square\diamond\left(r_i.running=true\right)$。

3.4 本章小结

本章首先使用数学定义方式，分别描述了空间位置结点，机器人移动的三个阶段观察、计算和移动。将机器人移动的三个阶段是否有原子性，以及机器人是否同步移动为调度模型的划分依据，介绍了三种调度模型完全同步调度模型、半同步调度模型和异步调度模型。介绍了机器人视觉快照表达式，F-R 视觉快照表达式更加灵活，且表达能力更强。移动机器人算法是以视觉快照为依据，所以移

动机器人算法是以视觉快照和对应的预设移动组合的式子，后续由以最小移动算法为例，讲解了稳定阶段和收敛阶段。自主移动机器人的核心是机器人协作完成预设任务，所以介绍了探索终止和永恒探索两种任务问题，并使用 LTL 公式具体描述了其满足的性质。

第四章　永恒探索算法建模

4.1　基础模型的构建

4.1.1　机器人建模

4.1.2　移动算法建模

4.1.3　任意初始位置的定义

4.2　调度模型的建模

4.3　本章小结

第五章　永恒探索算法符号模型验证

5.1　机器人初始位置定义

5.2　完全同步调度模型

5.2.1　非碰撞性

5.2.2　非碰撞性

5.2.3　非互换性

5.3　半同步调度模型

5.3.1　非碰撞性

5.3.2　非碰撞性

5.3.3　非互换性

5.4　完全异步调度模型

5.4.1　非碰撞性

5.4.2　非碰撞性

5.4.3　非互换性

5.5　实验结果分析

5.6　本章小结

第六章　总结和展望

6.1　论文总结

6.2　工作展望

致　　谢

蔡晓伟

二零一七年十一月

攻读硕士学位期间发表论文和科研情况

■ 已公开发表论文