

---

---

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

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
-	-	-	-
-	-	-	-
-	-	-	-

# 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

In recent years, with the rapid developments of society and science, the role that autonomous mobile robots play in people's daily life .It has a wide range of applications in some areas,such as electronics industry、 aerospace、 automotive production and so on.As human activity space.With the expansion of human activity space,started to hope that autonomous mobile robots can explore unknown space in place of human.researchers envision groups of mobile robots self-organizing and cooperating to solving some predefined tasks,including exploration of unknown environments,without any central coordinating authority.

Exploration Algorithm is the core problem for the autonomous mobile robot to complete exploration task,the movement of robot depends on the exploration algorithm.In the past,the vast majority of research on mobile robots considers that mobile robots are locating in a continuous two-dimensional Euclidian space. A recent trend was to the discrete model take the place of the classical continuous mode. In the discrete model,there exist three different scheduling modes called full synchronous model(FSYNC), semi-synchronous model(SSYNC) and asynchronous model(ASYNC).There exist many research finding,such as the Min-Algorithm、 the Max-Algorithm and so on,depending on handwritten proofs,which, particularly in asynchronous execution models,are both cumbersome and error-prone.

Formal methods for the automatic, rigorous and efficient has been used to verify the correctness of mobile autonomous robots algorithm。 for example, using DiVinE and ITS tools to model check mobile robot protocols.and applying Maude to model the Min-Algorithm.

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 coincides 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 . . . . .	5
1.4 论文结构 . . . . .	5
第二章 nuXmv 介绍 . . . . .	7
2.1 nuXmv 基本数据类型 . . . . .	8
2.2 nuXmv 有限状态机 . . . . .	9
2.2.1 变量声明 . . . . .	9
2.2.2 约束定义 . . . . .	11
2.2.3 模块声明 . . . . .	13
2.2.4 模块实例 . . . . .	14
2.3 LTL 模型检测 . . . . .	16
2.3.1 LTL 规范说明 . . . . .	16
2.3.2 LTL 验证方式 . . . . .	18
2.4 本章小结 . . . . .	18
第三章 永恒探索算法 . . . . .	20
3.1 机器人移动三个阶段 . . . . .	21
3.2 机器人移动调度策略 . . . . .	21
3.3 环形空间探索算法 . . . . .	24
3.3.1 机器人视觉快照 . . . . .	24

3.3.2	机器人移动算法 . . . . .	26
3.4	探索终止和永恒探索 . . . . .	29
3.4.1	探索终止 . . . . .	29
3.4.2	永恒探索 . . . . .	30
3.5	本章小结 . . . . .	30
第四章	永恒探索算法建模 . . . . .	32
4.1	机器人建模 . . . . .	32
4.1.1	移动算法匹配 . . . . .	33
4.1.2	机器人移动 . . . . .	36
4.2	调度策略建模 . . . . .	37
4.3	永恒探索性质 . . . . .	38
4.4	本章小结 . . . . .	40
第五章	自主移动机器人永恒探索算法验证 . . . . .	41
5.1	验证结果与分析 . . . . .	41
5.2	本章小结 . . . . .	43
第六章	总结和展望 . . . . .	44
6.1	总结 . . . . .	44
6.2	展望 . . . . .	44
参考文献	. . . . .	46
致谢	. . . . .	52
发表论文和科研情况	. . . . .	52

# 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, Stanford Research Institute) 的尼尔斯·尼尔森和查尔斯·罗森 (Charles Rosen) 等人，在 1966 年至 1972 年研发出了命名为 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 etc 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), 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

自主移动机器人空间探索 [1][2] 是机器人领域的一个研究热点，机器人之间协作完成空间探索任务的核心是机器人移动算法 [1][3][4][5] 的设定。机器人移动算法决定自主移动机器人的移动行为，从而实现对空间的探索。目前离散空间模型中主要研究机器人探索问题两种探索方式：探索终止 [6][7][8] 和永恒探索 [9]。而大多数研究者都将研究重点放在永恒探索上。

类似于网络拓扑结构，离散空间模型的结构也有多种，例如树形空间、环形空间、星型空间。文献 [8] 描述树形空间位置结点数为  $n$  时，需要  $\Omega(n)$  个机器人并且树的深度最大为 3 时，才可能实现永恒探索。文献 [7] 在环形空间位置结点数是  $n$  的情况下，需要机器人数量  $k = \iota(\log n)$  并且机器人数量  $k$  和环形空间位置结点数  $n$  互质时，才可能满足永恒探索性质。文献 [5] 提出满足永恒探索性质的最小移动算法和最大移动算法，最小移动算法是指环形空间中，机器人数量最少为  $k=3$ ，空间位置结点数至少为  $n=10$ ，且  $n$  与  $k$  的数量关系互质，并给出了证明当机器人数量为  $k=3$  时，空间位置结点数  $n$  小于 10 时，没有移动协议满足永恒探索性质。最大移动算法是环形空间位置结点数  $n$ ，机器人数量  $k=n-5$  且  $n$  与  $k$  的数量关系互质时，才可能满足永恒探索形式。在文献 [5] 中也同时给出了最小移动算法和最大

移动算法的实例，并利用手动推演的方式，进行了证明。文献 [10][11] 手动推演证明了网格空间的探索性质。

文献 [13] 和 [14] 探讨了机器人移动算法自动化验证的可能性。文献 [13] 使用 LUSTRE[15] 对  $3 \times 3$  网格空间永恒探索性质进行了描述，通过穷尽搜索的方式验证 3 个机器人在半同步调度策略下，没有搜索到满足永恒探索性质的移动算法。文献 [14] 中描述环形空间大小为  $n$ ，机器人数量为  $k$  时的永恒探索算法，并自动生成在半同步调度策略下的移动算法。

在采用形式化自动化验证方面，文献 [16] 首先使用形式化模型描述离散空间中机器人移动算法，并将形式化模型转化成 DiVinE 模型验证器 [17] 和 ITS 工具 [18] 支持的 DVE 语言，实现了移动算法的自动化验证。文献 [19] 则使用基于重写逻辑 Maude 形式化语言对移动算法进行建模，并使用 LTL 描述移动算法的永恒探索性和互排斥性，通过 Maude LTL 模型验证器 [20][21][22] 实现模型的验证。

上述对永恒探索算法形式化建模、验证和分析工作，相对于之前手动验证机器人移动算法效率上得到可巨大的提升。但是受限于形式化工具自身的形式，通过人为设定一个具体的初始状态进行验证，然而机器人移动算法的机器人初始位置状态是任意的。尽管从理论上是可以穷举所有机器人可能的初始位置状态对机器人移动算法进行一一验证，然而而在实际操作中不仅效率低下并容易发生遗漏而导致验证不完整。Aminof 等人提出一种参数化的模型检测方法用于聚集系统 (rendezvous systems)[23]。然而由于参数化模型检测的不可判定性 [24]，该方法只能用于某类特殊模型的验证。通常需要通过抽象 (abstraction) 或者归纳 (induction) 等手段对模型进行适当的转化使其模型检测问题变的可判定 [25][26]。

nuXmv[27] 是一种符号模型验证工具，具有高效的性能，可以对有限状态系统和无限状态系统进行建模与分析。利用 nuXmv 工具分别对机器人自主空间探索算法进行建模并利用 LTL 公式对机器人的移动行为进行定义。利用 nuXmv 提供的基于 BDD 方法 [28] 及 SMT 方法 [28] 实现了协议的符号模型检测, 验证移动算法是否满足永恒探索的性质。

### 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、机器人数量等相关信息。

### 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, Linear Temporal 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 word*[3]) 代表无符号 3 位 (bit) 字, (*signed word*[7]) 代表有符号 7 位 (bit) 字。通用无符号表达式为 (*unsigned word*[ $N$ ]) 和有符号表达式为 (*signed 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类型。

#### 变量类型定义

变量类型定义的语法如下：

```

1 type_specifier ::
2     simple_type_specifier
3     | module_type_specifier
4
5 simple_type_specifier ::
6     boolean
7     | word [ basic_expr ]
8     | unsigned word [ basic_expr ]
9     | signed word [ basic_expr ]
10    | real
11    | integer
12    | { enumeration_type_body }
13    | basic_expr .. basic_expr
14    | array basic_expr .. basic_expr
15    of simple_type_specifier
16
17 enumeration_type_body ::
18     enumeration_type_value
19     | 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 ;

```

输入变量同状态变量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：

```
1 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：

```
1 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：

```
1 VAR a : boolean;
2 FROZENVAR b : 0..1;
3 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：

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

```
1 VAR g : unsigned word[3];
2 VAR h : word[3];
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。

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

变量 **j** 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_constraint :: 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:

```

1  ASSIGN a := exp;      is equivalent to INVAR a in exp;
2  ASSIGN init(a) := exp; is equivalent to INIT a in exp;
3  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:

```

1  fairness_constraint ::
2      FAIRNESS simple_expr [:]
3      | JUSTICE simple_expr [:]
4      | 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:

```

1  module :: MODULE identifier [( module_parameters )] [module_body]
2
3  module_parameters ::
4      | module_parameters , identifier
5
6  module_body ::
7      module_element
8      | module_body module_element
9
10 module_body ::
11     module_element
12     | module_body module_element
13 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：



```

1 MODULE main
2 VAR
3   request : boolean;
4   state : {ready, busy};
5 ASSIGN
6   init(state) := ready;
7   next(state) := case
8     state = ready & request = TRUE : busy;
9     TRUE                           : {ready, busy};
10  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：

```

1 MODULE inverter(input)
2   VAR
3     output : boolean;
4   ASSIGN
5     init(output) := FALSE;
6     next(output) := !input;
7
8 MODULE main
9   VAR
10    gate1 : process inverter(gate3.output);
11    gate2 : process inverter(gate1.output);
12    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：

```
1 FAIRNESS
2   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 规范说明

有限状态机的建模与验证，其描述规格说明可以两个部分：一个是系统模型构建规范，另一个是验证性质规范。系统模型构建规范是通过模块定义和状态迁移实现，而验证性质规范是通过时序逻辑进行描述。nuXmv 的线性时序逻辑 (LTL) 验证器基于状态机初始状态和 LTL 公式状态性质描述之间的逻辑关系，给出模型验证的结果。

nuXmv 在模块定义中使用关键词 LTLSPEC 声明 LTL 规范，用于对预验证性质的原子命题的描述。LTL 规范的语法如下：

```
1 ltl_specification :: LTLSPEC ltl_expr [:]
2                   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`。

```

1 MODULE main
2   VAR
3     semaphore : boolean;
4     proc1 : process user(semaphore);
5     proc2 : process user(semaphore);
6   ASSIGN
7     init(semaphore) := FALSE;
8   LTLSPEC G ! (proc1.state = critical & proc2.state = critical)
9   LTLSPEC G (proc1.state = entering -> F proc1.state = critical)
10
11 MODULE user(semaphore)
12   VAR
13     state : {idle, entering, critical, exiting};
14   ASSIGN
15     init(state) := idle;
16     next(state) :=
17       case
18         state = idle : {idle, entering};
19         state = entering & !semaphore : critical;
20         state = critical : {critical, exiting};
21         state = exiting : idle;
22         TRUE : state;
23       esac;
24     next(semaphore) :=
25       case
26         state = entering : TRUE;
27         state = exiting : FALSE;
28         TRUE : semaphore;
29       esac;
30   FAIRNESS
31     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 验证方式

`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。

## 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 永恒探索算法

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。

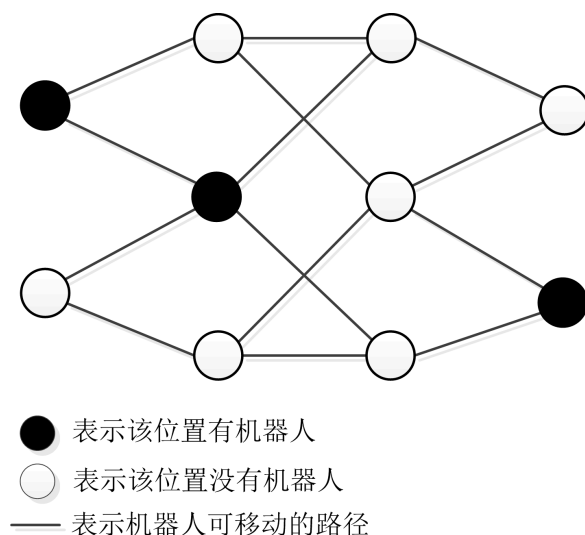


图 3.1: 无向连通图表示探索空间

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 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。

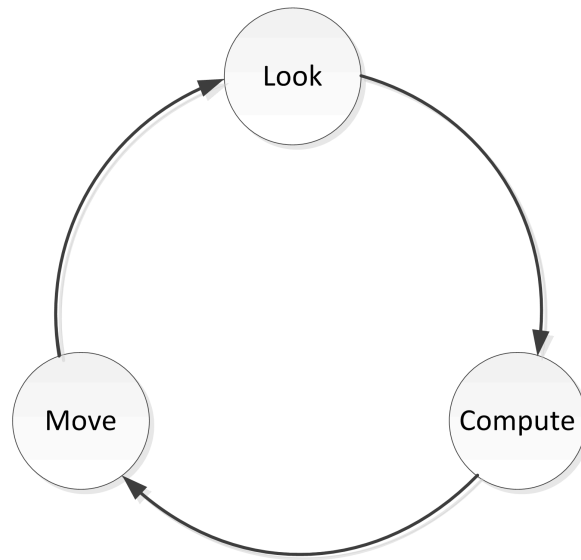


图 3.2: 机器人移动三阶段

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。

### 3.2 机器人移动调度策略

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  $Rob = \{r_1, r_2, r_3, \dots, r_k\}$  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  $Pos = \{n \in N^+ | 0, 1, 2, \dots, n-1\}$  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  $p : \{Rob \rightarrow Pos\}$  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置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有较大区别, 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)
2   while
3     choose Sched from Rob
4     synchronous {
5       foreach r in Sched{
6         r.look
7         r.compute
8         r.move
9       }
10    }

```

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:

```

1 FSYNC-SCHEDULE(Rob)
2   while
3     synchronous {
4       foreach r in Sched{
5         r.look
6         r.compute
7         r.move
8       }
9     }

```

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, 其他过程完全相同。

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, 所以可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。

```

1 ASYNC-SCHEDULE(Rob)
2   asynchronous {
3     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。

### 3.3 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。

#### 3.3.1 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

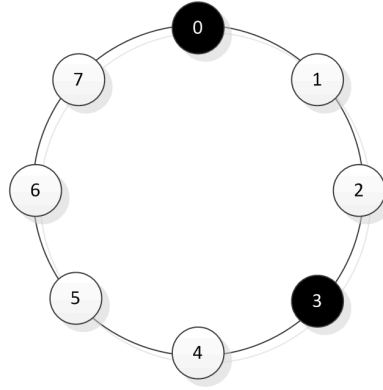


图 3.3: 环形拓扑结构探索空间

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  $d_j = 0$ 。

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  $\delta_p^F$ ，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：  $\delta_p^+ = \langle d_j, d_{j+1}, \dots, d_{j+n-1} \rangle$ 。

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

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 Ielia.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。

### 3.3.2 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 phase*) 和收敛阶段 (*Convergence phase*)。

表 3.1: 最小移动算法稳定阶段

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

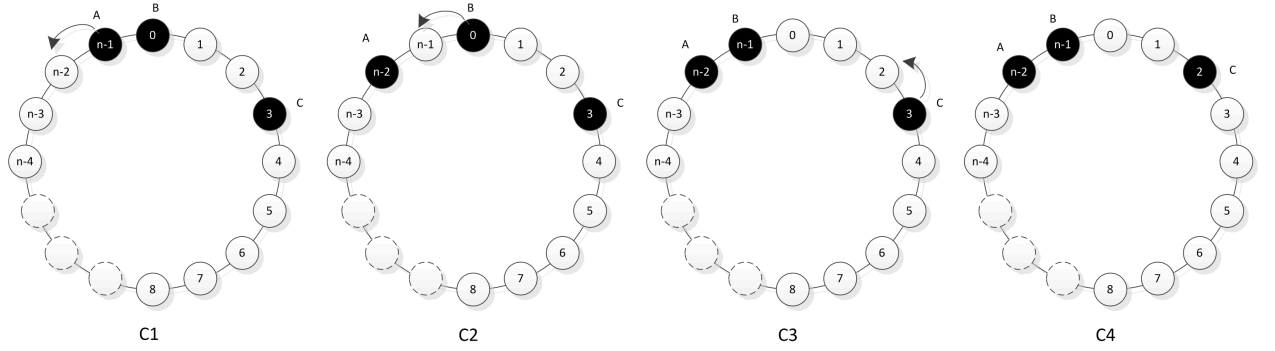


图 3.4: 最小移动算法的稳定阶段

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

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

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。

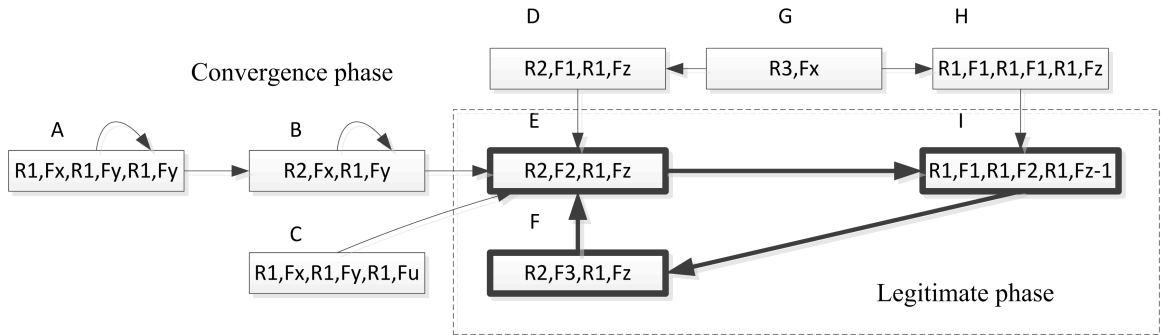


图 3.5: 最小机器人移动算法状态转移

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。

### 3.4 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  $n$  和机器人数量  $k$ , 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, 在机器人数量  $k \geq 17$ , 机器人数量  $k$  与结点数  $n$  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。

#### 3.4.1 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):**  $\bigwedge_{j=0}^{n-1} \Diamond (d_j > 0)$ 。

### 3.4.2 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):**  $\bigwedge_{i=1}^k \bigwedge_{j=1}^k \Box (i \neq j \rightarrow c_{r_i} \neq c_{r_j})$ 。

**非互换性 (No switch):**  $\bigwedge_{h=0}^{n-1} \bigwedge_{i=1}^k \bigwedge_{j=1}^k \Box (c(r_i) = h \wedge c(r_j) = (h+1) \bmod n \rightarrow (\neg (c(r_i) = (h+1) \bmod n) \wedge c(r_j) = h))$ 。

**非终止性 (Live):**  $\bigwedge_{h=0}^{n-1} \bigwedge_{i=1}^k \Box \Diamond (c(r_i) = h)$ 。

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):**  $\bigwedge_{i=1}^k \Box \Diamond (r_i.running = true)$ 。

## 3.5 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

判定移动机器人移动算法为永恒探索算法的核心，是验证其在公平性先决条件下是否同时满足永恒探索算法的三条性质：非冲撞性、非互换性和非终止性。将机器人和探索空间视为一个完整的系统，在这个系统中不仅包括机器人和空间位置，而且包括机器人移动算法和调度模型。使用 `nuXmv` 对系统中空间位置、机器人性质、移动算法以及不同调度进行抽象，从而模拟移动算法中的状态迁移。本文中使用的空间模型是离散空间模型，不仅简化了模型的构建，而且将建模的核心放在了移动算法和机器人性质的构建上。考虑到不同拓扑结构的离散空间有不同的性质和特点，从而对于不同的结构的离散空间，会根据其自身特点进行模型的定义。为了更加具体说明永恒探索算法的建模与验证过程，本文以验证环形空间上最小移动算法是否是永恒探索算法为例，介绍符号模型检验方法在永恒探索算法中的应用思想。

### 4.1 机器人建模

利用 `nuXmv` 提供的参数化模块 (`parameterized module`) 对机器人进行建模，参数化模块在实例化模块是可以给模块实例传入所有机器人位置编号变量。在机器人模块中定义变量 `phase` 表示机器人所处的操作阶段。机器人中观察 (`Look`) 和计算 (`Compute`) 在模型中可以合并为一个操作，称作观察计算，符号定义为 `lc(LookCompute)`，因为这两个操作只是做出了一个移动决策。机器人处于移动操作使用符号 `m` 表示，那么 `phase` 变量只有两种取值 `lc` 和 `m`。机器人在观察移动操作做出的移动决策可以使用移动变量 `move` 表示，在环形离散空间模型中，`move`

有三种取值 -1、0 和 1，-1 表示机器人逆时针移动，0 表示机器人不发生移动，1 表示机器人顺时针移动。

```

1 MODULE robot(p1,p2,...)
2   VAR
3     phase : {lc,m};
4     move : -1..1;
5     ...

```

上述声明的是机器人模块，模块名称为 **robot**，并且模块带有位置环形空间所有机器人位置传入参数  $p1, p2, \dots$ 。在机器人模块实例化时，参数列表第一个位置值  $p1$  赋值为当前实例机器人在环形空间中的位置值，以当前机器人位置为起点，沿着环形空间顺时针方向依次是  $p2$ 、 $p3$ ...，即参数列表是以当前实例机器人的位置为起点，沿着环形空间顺时针方向上的机器人依次传入。

#### 4.1.1 移动算法匹配

机器人的移动算法是机器人在计算操作时，作为移动决策的判断依据。在 **nuXmv** 建模过程采用顺时针或者逆时针相邻两个机器人的之间的没有机器人空间位置结点的个数作为匹配依据。相邻机器人 A 到机器人 B 之间没有机器人空间位置结点个数使用  $gap_{A \rightarrow B}^F$  表示，其中  $F \in \{+, -\}$  表示  $gap$  的计算取值方向，+ 表示顺时针，- 表示逆时针。

$gap_{A \rightarrow B}^F$  的计算公式如下：

$$gap_{A \rightarrow B}^+ = (c(B) - c(A) + n) \bmod n$$

$$gap_{A \rightarrow B}^- = (n - ((c(B) - c(A) + n) \bmod n)) \bmod n$$

计算公式中  $n$  表示环形空间位置结点个数。机器人在环形空间中任意一个时刻都有占据一个空间位置。如图4.1中所示，环形空间结点的位置是从数字 0 开始验证环形空间顺时针方向依次递增编号，最后一个编号为  $n-1$ ，其中  $n$  是表示环形空间中位置结点的个数。图中顺时针 1 和顺时针 2、逆时针 1 和逆时针 2 可以看

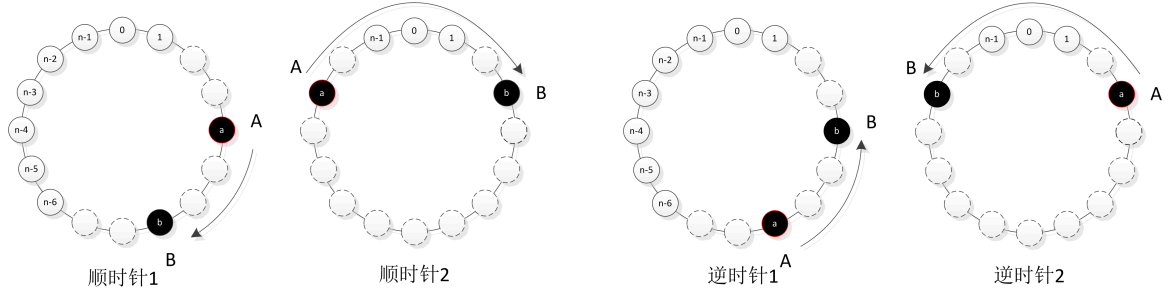


图 4.1: 机器人 A 和 B 之间顺时针和逆时针 gap 计算示意图

出，当顺时针和逆时针计算机器人 A 到机器人 B 的  $gap_{A \rightarrow B}^F$  时，机器人 A 和机器人 B 之间包含位置结点为 0 的情况。机器人 B 与机器人 A 之间的顺时针间隔为两者节点位置的差加  $n$  再以  $n$  为基数取模。逆时针的间隔则为  $n$  减去顺时针的间隔得到的结果再以  $n$  为基数取模。之所以继续取模是考虑 A 与 B 位置重叠的情况，此时，顺时针与逆时针方向的间隔均为 0。

假设机器人  $r$  的快照 F-R 视觉快照  $\delta_{c(r)}^F$  满足移动算法中的某条规则  $L$ ，记为  $match(\delta_{c(r)}^F, L)$ 。其中规则  $L$  可确定的移动方向使用符号  $\beta(L)$  表示。机器人在每次计算阶段之后，会获取一个移动决策前进 (Front)，后退 (Back) 或者保持不动 (Idle)，那么分别对应的移动量为 1, -1, 0。所以机器人  $r$  的移动决策可以使用其移动量  $M_{(r)}$  表示，具体计算公式如下：

$$M_{(r)} = \begin{cases} 1 & \text{if } \left( match(\delta_{c(r)}^F, L) \wedge F = + \wedge \beta(L) = Front \right) \vee \\ & \left( match(\delta_{c(r)}^F, L) \wedge F = - \wedge \beta(L) = Back \right) \\ -1 & \text{if } \left( match(\delta_{c(r)}^F, L) \wedge F = - \wedge \beta(L) = Front \right) \vee \\ & \left( match(\delta_{c(r)}^F, L) \wedge F = + \wedge \beta(L) = Back \right) \\ 0 & \text{otherwise} \end{cases}$$

对于环形空间，若机器人  $r$  在空间位置  $c(r)$  的顺时针和逆时针快照相同  $\delta_{c(r)}^+ = c(r)^-$ ，也就是说机器人  $r$  的快照是对称的。若此时机器人  $r$  的快照匹配移动算法中某条规则  $L$ ，根据规则  $L$  得出的移动方向  $\beta(L)$  无论是前进或者后退，即机器人的移动量可以是 1 或者 -1，此时机器人会随机选择其中一个作为最终

的移动决策。若 $r$ 的为快照不匹配算法当中的任意一个移动规则时移动量就为 $M_{(r)} = 0$ 。

下面代码描述环形空间的节点数为 $n=10$ ，机器人个数为 $k=3$ 时的最小移动算法。 $next(move)$ 是机器人 $r$ 的移动量 $M_{(r)}$ 。 $phase = lc$ 是移动规则匹配的前置条件，当机器人处于观察计算阶段时，才会匹配移动算法。

```

1  next(move) :=      --变量 move 在下一步的值
2  case
3    phase = lc & ((p2 - p1 + 10) mod 10 = 1) & ((p3 - p2 + 10) mod 10 = 3) : -1;
4    --顺时针RC1
5    phase = lc & ((10 - ((p3 - p1 + 10) mod 10)) mod 10 = 1) & ((10 - (p2 - p3 + 10) mod 10) mod 10
6    = 3) : 1;
7    --逆时针RC1
8    ...
9    phase = lc & ((p2 - p1 + 10) mod 10 >= 1) & (((p2 - p1 + 10) mod 10) = ((p1 - p3 + 10) mod 10)
10   ) & ((p3 - p2 + 10) mod 10 != (p2 - p1 + 10) mod 10) : {1,-1};
11   --顺时针RC2
12   phase = lc & ((10 - ((p3 - p1 + 10) mod 10)) mod 10 >= 1) & ((10 - ((p3 - p1 + 10) mod 10)) mod
13   10 = (10 - ((p1 - p2 + 10) mod 10)) mod 10) & ((10 - (p2 - p3 + 10) mod 10) mod 10 != (10
14   - ((p3 - p1 + 10) mod 10)) mod 10) : {1,-1};
15   --逆时针RC2
16   ...
17   TRUE : 0;
18   --other
19   esac;

```

代码 3-4 行描述规则 RC1 的顺时针方向的匹配， $p1$  表示当前机器人实例的自身位置，在  $p1$  位置上的机器人  $r$  从自身开始，按照顺时针方向，依次计算  $p1$  到  $p2$ 、 $p2$  到  $p3$  之间的间隔个数。对应匹配 RL1 中机器人的之间的间隔个数，以此作为机器人  $r$  顺时针匹配移动算法的依据，当满足匹配时 $next(move)$  的取值为 -1。代码 5-6 行描述 RC1 的逆时针方向的匹配，逆时针方向的匹配则是计算  $p1$  到  $p3$ 、 $p3$  到  $p2$  之间的间隔个数作为匹配依据，当满足匹配时 $next(move)$  的取值为 1。代码 9-12 行描述规则 RC2 的顺时针和逆时针匹配，由于 RC2 规则描述的是对称情况下的移动规则，对于匹配该规则机器人的移动量可能是 -1 或者 1。

### 4.1.2 机器人移动

机器人经过观察计算阶段之后，会进入移动阶段。机器人在观察计算和移动两种阶段之间切换。

```

1 next(phase) :=
2   case
3     phase = lc    : m;
4     phase = m     : lc;
5     TRUE          : phase;
6   esac;

```

在 nuXmv 建模过程中，机器人移动阶段状态分为两个：观察计算，符号 *lc* 表示；移动，符号 *m* 表示。下一个移动阶段状态 *next(phase)* 取决于当前移动阶段状态 *phase*：当当前阶段状态 *phase* 为 *lc* 下一个移动阶段状态 *next(phase)* 的值为 *m*；当当前阶段状态 *phase* 为 *m*，下一个移动阶段状态 *next(phase)* 的值为观察计算阶段 *lc*；

当机器人移动阶段状态为移动 *m* 时，其下一个位置结点编号为  $next(c(r)) = c(r)$ 。机器人移动至新的空间位置上，改变了整个空间中位置快照，所有机器人会根据新的位置快照，做出对应的移动决策。

```

1 next(p1) :=
2   case
3     phase = m & (move + p1) < 0          : 10 + (move + p1);
4     phase = m & (move + p1) >= 0 & (move + p1) < 10 : (move + p1);
5     phase = m & (move + p1) >= 10         : (move + p1) - 10;
6     TRUE                                  : p1;
7   esac;

```

*next(p1)* 表示当前模块实例化的机器人下一个位置，这个过程只在机器人当前移动阶段状态 *phase* 为 *m* 时才会进行。考虑到空间位置范围问题，当机器人顺时针和逆时针通过位置为 0 的结点时，都会对新的空间位置值进行对应的调整。将机器人的新的位置计算过程归纳如下：

$$next(p1) = \begin{cases} (move + p1) + n & if (move + p1) < 0 \\ (move + p1) & if (move + p1) \geq 0 \wedge (move + p1) < n \\ (move + p1) - n & if (move + p1) \geq n \end{cases}$$

## 4.2 调度策略建模

nuXmv 模块化建模的优点是代码可移植性强, 在完全同步调度策略、半同步调度策略和完全异步调度策略进行建模时, 主模块和机器人模块的建模代码可以复用。三种不同的调度策略情况下, 只需根据不同的调度特有性质进行建模调整。

**半同步调度策略建模** 对于半同步调度策略, 每个机器人在调度过程中只有选中和非选中两种情况。在机器人模型中定义调度变量dispatcher, 变量声明如下:

```
1 VAR dispatcher : {choose,steady};
```

使用关键字VAR 定义符号枚举类型变量dispatcher, 表示机器人的调度标识。其枚举元素有choose 和steady, 分别表示机器人被选中和没被选中。在半同步调度模型中, 机器人的每个移动过程中, 是随机调度的。机器人随机调度建模如下:

```
1 next(dispatcher) :=
2   case
3     TRUE          : {choose,steady};
4   esac;
```

建模代码中机器人下一个调度标识只有一个永真的条件分支, 并且取值是{choose,steady}, 即机器人每次移动过程中下一个调度标识是choose 和steady 当中随机取值。机器人在匹配移动算法时, 若 dispatcher 为 choose, 才进行移动算法的匹配。

```
1 next(move) :=      -- 变量 move 在下一步的值
2   case
3     dispatcher = choose & phase = lc & ((p2 - p1 + 10) mod 10 = 1) & ((p3 - p2 + 10) mod 10 = 3) :
4       -1;
5     -- 顺时针RCI
6     dispatcher = choose & phase = lc & ((10 - ((p3 - p1 + 10) mod 10)) mod 10 = 1) & ((10 - (p2 - p3
7       + 10) mod 10) mod 10 = 3) : 1;
8     -- 逆时针RCI
9     ...
10    TRUE : 0;
11  esac;
```

移动算法建模中, 对每个移动规则匹配添加先决条件dispatcher = choose, 表明机器人只有在选中是才会计算移动量, 否则移动量永远为 0。

**完全同步调度策略建模** 完全同步调度策略是半同步调度策略的一种特殊情况，空间中所有机器人在每次移动过程中都会被调度。在半同步调度策略建模的基础上，调整模型中调度标识的值永远是选中choose:

```
1 next(dispatcher) :=
2   case
3     TRUE      : choose;
4   esac;
```

**完全异步调度策略建模** 完全异步调度策略中所有机器人的移动都是并行的，在 nuXmv 中实例化机器人模型时，通过关键字process 声明实例是异步的。

```
1 VAR
2   r1 : process robot(pos1,pos2,pos3);
3   r2 : process robot(pos2,pos3,pos1);
4   r3 : process robot(pos3,pos1,pos2);
```

实例化三个异步机器人 r1、r2、r3，在模型验证的过程中，为了确保机器人公平获得执行时间片，可以在机器人模块定义中添加公平性约束。

```
1 FAIRNESS
2   running
```

### 4.3 永恒探索性质

判定移动算法是否是永恒探索算法，需要移动算法满足三条性质：非冲撞性、非互换性和非终止性。下面以环形空间位置数为 10，机器人数为 3 的最小移动算法为例，任意机器人在任意初始位置，使用 LTL 公式描述永恒探索算法的三条性质并验证模型对性质的满足性。主模块中声明三个机器人位置变量 pos1、pos2、pos3，作为全局变量记录三个机器人的位置。建模时设定三个机器人位置排列顺序是按照顺时针方向依次排列 pos1、pos2、pos3。

```
1 MODULE main
2   VAR
3     pos1 : 0..9;
4     pos2 : 0..9;
5     pos3 : 0..9;
6     r1 : robot(pos1,pos2,pos3);
7     r2 : robot(pos2,pos3,pos1);
8     r3 : robot(pos3,pos1,pos2);
```



环形空间结点数为  $n$  时, 空间位置结点编号为  $0 \dots n-1$ 。当  $n=10$  时, 机器人的位置编号范围是  $0$  到  $9$ 。在机器人模块实例化时, 传入参数列表的第一个参数为当前机器人实例的位置值, 后续依次是顺时针方向上机器人位置。例如机器人  $r1$  实例化中, 第一位参数  $pos1$ , 表示当前实例是  $pos1$  位置上的机器人, 并且后续是以  $pos1$  为起点顺时针方向依次是  $pos2$  和  $pos3$ , 进行位置参数传入。

使用 LTL 定义机器人初始位置时, 只需限制每个机器人位置的范围和顺时针排列次序。

```
1 (((pos3 > pos2) & (pos2 > pos1)) | ((pos1 > pos3) & (pos3 > pos2)) | ((pos2 > pos1) & (pos1 > pos3)))
```

按照顺时针方向,  $pos1$ 、 $pos2$ 、 $pos3$  的大小次序有三种可能性:1、 $pos1 < pos2 < pos3$ ;2、 $pos2 < pos3 < pos1$ ;3、 $pos3 < pos1 < pos2$ ; 在 nuXmv 中未给变量分配初始值时, 会对该变量的所有取值可能性进行穷举验证。该初始化定义可以实现机器人初始位置任意性的描述。

**非冲撞性** 非冲撞性是指离散空间模型中, 同一时刻同一位置结点上至多只有一个机器人, 使用 LTL 描述就是同一时刻任意两个机器人所在位置不能相同, 如下:

```
1 LTLSPEC (((pos3 > pos2) & (pos2 > pos1)) | ((pos1 > pos3) & (pos3 > pos2)) | ((pos2 > pos1) & (pos1 > pos3))) -> (G ((pos1 != pos2) & (pos2 != pos3) & (pos1 != pos3)))
```

上述 LTL 定义中, 三个机器人位置  $pos1$ 、 $pos2$  和  $pos3$  的初始化位置是任意的。nuXmv 中 LTL 符号  $G$  表示在将来所有时刻都满足一定条件, 在非冲撞性中, 表示将来所有时刻都满足  $pos1$ 、 $pos2$  和  $pos3$  中任意两个位置都不都相同。

**非互换性** 非互换性是指非互换性对应的 LTL 公式表示如果两个机器人  $r_i$  与  $r_j$  物理上相邻, 即其所在节点的编号相差 1, 则一定不会出现  $r_i$  与  $r_j$  在下一个状态位置互换的情况。

```
1 LTLSPEC (((pos3 > pos2) & (pos2 > pos1)) | ((pos1 > pos3) & (pos3 > pos2)) | ((pos2 > pos1) & (pos1 > pos3))) -> G (((pos1 + 1) mod 10) = pos2 -> (X (((pos2+1) mod 10) != pos1)) & (((pos2 + 1) mod 10) = pos3) -> (X (((pos3+1) mod 10) != pos2)) & (((pos3 + 1) mod 10) = pos1) -> (X (((pos1+1) mod 10) != pos3)))
```

非互换性中机器人初始位置定义与非冲撞性中的相同。nuXmv 中 LTL 符号  $X$  表示在下一个时刻都满足一定条件。由于先决条件是机器人位置  $pos1$ 、 $pos2$  和  $pos3$  是在环形空间中是顺时针方向获取的，若当前机器人位置加上 1 后取模等于前面的机器人位置，那么说明它们相邻，在下一个时刻前面的机器人位置加上 1 后取模等于前面的机器人位置，表明两者交换了位置。例如  $((pos1 + 1) \bmod 10)$  等于前面位置上的机器人位置  $pos2$ ，表示  $pos1$  与  $pos2$  相邻，下一个时刻  $pos2+1$  取模不能等于  $pos1$ ，即  $pos1$  和  $pos2$  位置不能互换。

**非终止性** 非终止性离散空间中每个机器人对每个空间位置结点进行反复的巡查。

```
1 LTLSPEC (((pos3 > pos2) & (pos2 > pos1)) | ((pos1 > pos3) & (pos3 > pos2)) | ((pos2 > pos1) & (pos1 > pos3))) -> (((G F (pos1 = 0)) & (G F (pos1 = 1)) &...& (G F (pos1 = 9))) & ((G F (pos2 = 0)) & (G F (pos2 = 1)) &...& (G F (pos2 = 9))) & ((G F (pos2 = 0)) & (G F (pos2 = 1)) &...& (G F (pos2 = 9))))
```

命题  $p$  描述机器人的位置取值，例如  $pos1=1$ ， $pos1=2$  等等。LTL 公式  $G F p$  等价于  $G (F p)$ ， $F p$  表示将来的某个时刻  $p$  为真， $G (F p)$  表示将来  $(F p)$  一直为真，即  $p$  在未来的某些时刻为真。在此表示机器人对某个空间结点反复巡查。

## 4.4 本章小结

本章以最小移动算法为例，详细介绍了使用 nuXmv 对机器人模块定义包括移动算法建模、移动算法的匹配。根据三种不同调度策略的性质，对建模做了对应调整。使用 LTL 公式描述永恒探索算法的三条性质：非冲撞性、非互换性和非终止性。通过模型真实的模拟了环形空间中机器人的移动行为，定义了永恒探索算法的性质。

## 第五章 自主移动机器人永恒探索算法验证

本文通过 nuXmv 符号模型检测工具对自主移动机器人永恒探索算法进行建模。通过参数化模块定义了机器人模块，包括移动算法、移动算法匹配和移动决策的执行。目前存在三种机器人调度策略，所以针对三种调度策略对机器人模块分别进行了对应的模型调整。为了验证永恒探索探索算法的完备性、准确性和满足性，通过 LTL 公式对机器人的任意初始位置和永恒探索算法的性质进行了定义。

以最小移动算法为验证对象，在完全同步调度策略、半同步调度策略和完全异步调度策略三种调度策略下进行形式化建模，并分别验证移动算法是否满足非冲撞性、非互换性和非终止性。验证内容如表 5.1 所示。

表 5.1: 最小移动算法验证概览

调度策略	非碰撞性	非互换性	非终止性
完全同步调度策略	验证	验证	验证
半同步调度策略	验证	验证	验证
完全异步调度策略	验证	验证	验证

### 5.1 验证结果与分析

根据上一章节中定义的自主移动机器人空间永恒探索算法模型，利用 nuXmv 符号模型检测工具对最小移动算法非冲撞性、非互换性和非终止性在不同的调

度策略下进行模型检测。实验运行的系统环境为 Windows 7 旗舰版, 硬件环境为 CPU Intel Xeon(R) 3.40GHz, 16G 内存。图5.1给出了非碰撞性、非碰撞性和非互换性三个性质在不同机器人调度策略下的验证结果和验证所需时间。实验中考虑了节点个数分别为 10 到 17 的情况, 其中由于算法要求机器人个数与节点个数互质, 节点为 12 与 15 的情况在实验中略去。

节点数	完全同步调度模型						半同步调度模型						异步调度模型					
	非碰撞性		非终止性		非互换性		非碰撞性		非终止性		非互换性		非碰撞性		非终止性		非互换性	
	结果	耗时	结果	耗时	结果	耗时	结果	耗时	结果	耗时	结果	耗时	结果	耗时	结果	耗时	结果	耗时
10	✓	1.4s	✓	484.0s	✓	1.9s	✓	0.7s	✓	83.1s	✓	1.7s	✗	1.2s <sup>*</sup>	✗	14.7s <sup>*</sup>	✓	0.9s
11	✓	5.4s	✓	3181.1s	✓	4.3s	✓	1.1s	✓	723.1s	✓	4.3s	✗	9.3s <sup>*</sup>	✗	18.1s <sup>*</sup>	✓	0.7s
13	✓	8.3s	✓	42.4s <sup>*</sup>	✓	6.8s	✓	1.1s	✓	63.9s <sup>*</sup>	✓	6.7s	✗	10.7s <sup>*</sup>	✗	2.1s <sup>*</sup>	✓	0.6s
14	✓	8.9s	✓	90.4s <sup>*</sup>	✓	8.4s	✓	1.1s	✓	129.2s <sup>*</sup>	✓	8.8s	✗	9.7s <sup>*</sup>	✗	48.8s <sup>*</sup>	✓	0.6s
16	✓	1.2s	✓	168.4s <sup>*</sup>	✓	2.3s	✓	0.6s	✓	389.2s <sup>*</sup>	✓	2.4s	✗	4.9s <sup>*</sup>	✗	19.6s <sup>*</sup>	✓	0.7s
17	✓	13.0s	✓	315.7s <sup>*</sup>	✓	11.4s	✓	2.0s	✓	562.3s <sup>*</sup>	✓	12.2s	✗	27.5s <sup>*</sup>	✗	102.2s <sup>*</sup>	✓	0.6s

注: ✓表示验证结果为真;✗表示验证结果为假;\*表示在设置初始状态的情况下验证所需时间;-表示超时;

※表示采用 SMT 方法验证所需时间;未标注的时间均表示采用 BDD 方法所需的验证时间。

图 5.1: 最小移动算法实验结果

实验结果表明最小移动算法在完全同步调度策略和半同步调度策略下, 满足对环形空间永恒探索性质, 即移动算法同时满足非碰撞性、非互换性和非终止性。而在异步调度模型下, 验证的结果为移动算法不满足非碰撞性与非终止性, 验证过程中 nuXmv 返回相应的反例。根据反例分析, 永恒探索性质不能被满足的原因是异步过程中机器人使用过时的快照信息做出的移动决策会导致相邻的机器人发生碰撞, 此时同一个位置结点上有两个机器人, 后续所有机器人没有与移动算法相匹配, 导致所有的机器人都不能移动。由此可见, 非终止性不被满足的原因是非碰撞性没有被满足。这一结果与 Ha 等人利用 Maude 模型检测得出的反例相同。在其反例中, 用于出现了两个机器人碰撞的情况而导致所有机器人无法移动, Ha 等人将此情况称为死锁状态。与 Ha 等人的验证不同的是其方法需要给出具体的初始状态才发现了反例, 而如何发现导致反例的初始状态文章并没有交待。本文提出的方法可以在不给出初始状态的前提下依然找到对应的反例。

nuXmv 支持基于 BDD 和 SMT 方法的验证。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, SMT 方法可以更快的找到反例。在实验中采用两种方法对三个性质进行验证。结果表明大部分验证都可以在相对较短的时间内完成。虽然随着空间节点数的增长验证所需的时间也会有所增加, 但依然可以在较合理的时间如一小时内完成。同 Béatrice 与 Ha 等人的工作相比, nuXmv 找到反例的时间更短。然而在验证被算法满足的性质时, nuXmv 所需的时间相对较长, 这是因为 nuXmv 不需要设定具体的初始状态, 因此其搜索的状态空间比固定初始状态时更大, 所需的时间则较长。实验数据中有部分实验是设定机器人初始状态的情况下进行的, 验证所需的时间明显缩短。

## 5.2 本章小结

本章详细列出了最小移动算法的永恒探索性质的实验结果, 从实验结果中可以得出符号模型对移动机器人空间永恒探索算法验证的高效性。

# 第六章 总结和展望

## 6.1 总结

本文使用 nuXmv 符号模型验证工具对自主移动机器人空间永恒探索算法进行建模与验证，对实验结果进行分析。使用参数模块定义了机器人模型，LTL 描述永恒探索的性质。通过使用 nuXmv 工具对最小移动算法的验证，提出了一种对空间永恒探索算法的验证方法。

本文中主要的工作包括以下几点：

第一，本文提出了机器人符号化建模方式，使用参数模块定义机器人模型，对移动算法及其匹配进行了描述。同时，使用 LTL 对空间永恒探索性质也进行了定义。

第二，本文在完全同步调度策略、半同步调度策略和完全异步调度策略下，实现了空间探索算法的模型构建，并使用 BDD 和 SMT 验证方式，分别验证模型的永恒探索性质。

第三，本文在已有自主移动机器人用探索算法形式化验证的基础上，实现了符号化模型的验证与分析。不仅提升的机器人移动算法的验证效率，而且扩展了验证功能。能够在任意机器人初始位置时，验证移动算法性质。

## 6.2 展望

本文主要是研究机器人空间探索问题。目前在使用形式化方法对机器人移动算法进行验证的研究工作并不是很多。本文也只是针对机器人空间探索问题的一小部分进行了研究。

机器人领域的研究十分广泛，除了空间探索之外，还有仿生学、网络机器人、多机器人系统等等。目前机器人领域的研究已经取得了很多成果，但是在某些方面离实用要求还是有些差距。随着传感器技术、机器学习和嵌入式技术的不断发展，机器人在未来人们生活和生产中逐渐扮演重要角色。

对于自主移动机器人空间探索问题，使用符号模型检测方法可以有效的避免状态爆炸问题，然而验证随着空间位置结点和机器人数量增加，符号模型的验证的时间也在快速增长。本文提出的符号模型验证方法，目前还不能直接用于实际系统中空间结点或机器人结点较多的应用场景。在符号模型验证方法的基础上，通过抽象或者归纳技术，如 `nuXmv` 内置的 `k-induction` 方法，解决验证效率问题，是未来需要进一步深入研究的工作。

## 参考文献

- [1] Flocchini P, Prencipe G, Santoro N. Distributed Computing by Oblivious Mobile Robots[J]. 2012, 3(2):1-185.
- [2] Almeida A, Ramalho G, Santana H, et al. Recent Advances on Multi-agent Patrolling[C]// Advances in Artificial Intelligence - Sbia 2004, Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings. DBLP, 2004:474-483.
- [3] Suzuki I, Yamashita M. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. Sirocco'96, the, International Colloquium on Structural Information Communication Complexity, 1999:313-330.
- [4] Suzuki I, Yamashita M. Erratum: Distributed anonymous mobile robots: formation of geometric patterns. Siam Journal on Computing, 1999, 28(4):1347-1363.
- [5] Blin L, Milani A, Potop-Butucaru M, et al. Exclusive Perpetual Ring Exploration without Chirality. Lecture Notes in Computer Science, 2010, 6343:312–327.
- [6] Devismes S, Petit F, Tixeuil S. Optimal Probabilistic Ring Exploration by Asynchronous Oblivious Robots[C]// International Conference on Structural Information and Communication Complexity. Springer-Verlag, 2009:195-208.
- [7] Flocchini P, Ilcinkas D, Pelc A, et al. Computing Without Communicating: Ring Ex-



- ploration by Asynchronous Oblivious Robots[M]// Principles of Distributed Systems. Springer Berlin Heidelberg, 2007:105-118.
- [8] Flocchini P, Ilcinkas D, Pelc A, et al. Remembering without memory: Tree exploration by asynchronous oblivious robots[C]// International Colloquium on Structural Information and Communication Complexity. Springer Berlin Heidelberg, 2008:33-47.
- [9] Baldoni R, Bonnet F, Milani A, et al. Brief Announcement: On the Solvability of Anonymous Partial Grids Exploration by Mobile Robots[C]// International Symposium on Distributed Computing. Springer-Verlag, 2008:496-497.
- [10] Baldoni R, Bonnet F, Milani A, et al. On the Solvability of Anonymous Partial Grids Exploration by Mobile Robots. Principles of Distributed Systems. Springer Berlin Heidelberg, 2008:428-445.
- [11] Devismes S, Lamani A, Petit F, et al. Optimal Grid Exploration by Asynchronous Oblivious Robots. Symposium on Self-Stabilizing Systems. Springer Berlin Heidelberg, 2012:64-76.
- [12] Suzuki I, Yamashita M. Erratum: "Distributed anonymous mobile robots: formation of geometric patterns" [SIAM J. Comput. 28 (1999), no. 4, 1347–1363 (electronic); MR1681010].[J]. Siam Journal on Computing, 2006, 36(4):279-280.
- [13] Devismes S, Lamani A, Petit F, et al. Optimal grid exploration by asynchronous oblivious robots[J]. Lecture Notes in Computer Science, 2011, 7596:64-76.
- [14] Potop-Butucaru M G, Potop-Butucaru M G. Brief announcement : discovering and assessing fine-grained metrics in robot networks protocols[C]// International Conference on Stabilization, Safety, and Security of Distributed Systems. Springer-Verlag, 2012:282-284.

- 
- [15] Halbwachs N, Caspi P, Raymond P, et al. The synchronous data flow programming language LUSTRE[J]. *Proceedings of the IEEE*, 2002, 79(9):1305-1320.
- [16] Bérard B, Lafourcade P, Millet L, et al. Formal verification of mobile robot protocols[J]. *Distributed Computing*, 2016:1-29.
- [17] Barnat J, Brim L, Havel V, et al. DiVinE 3.0 – An Explicit-State Model Checker for Multithreaded C C++ Programs[C]// *International Conference on Computer Aided Verification*. Springer-Verlag, 2013:863-868.
- [18] Colange M, Baarir S, Kordon F, et al. Towards Distributed Software Model-Checking Using Decision Diagrams[M]// *Computer Aided Verification*. Springer Berlin Heidelberg, 2013:830-845.
- [19] Doan H T T, Bonnet F, Ogata K. Model Checking of a Mobile Robots Perpetual Exploration Algorithm[C]// *International Workshop on Structured Object-Oriented Formal Language and Method*. Springer-Verlag New York, Inc. 2016:201-219.
- [20] Eker S, Meseguer J, Sridharanarayanan A. The Maude LTL Model Checker[J]. *Electronic Notes in Theoretical Computer Science*, 2004, 71(05):162-187.
- [21] Clavel M, Durán F, Eker S, et al. All About Maude - A High-Performance Logical Framework[J]. 2007, 4350.
- [22] Goranko V. Logic in Computer Science : Modelling and Reasoning About Systems[J]. *Journal of Logic Language Information*, 2007, 16(1):117-120.
- [23] Hagiya M, Mitchell J C. Proceedings of the International Conference on Theoretical Aspects of Computer Software[C]// *International Conference on Theoretical Aspects of Computer Software*. Springer-Verlag, 1994:257-271.

- [24] Apt K R, Kozen D C. Limits for Automatic Verification of Finite-State Concurrent Systems.[J]. Information Processing Letters, 1986, 22(6):307-309.
- [25] Clarke E M, Grumberg O, Jha S. Verifying Parameterized Networks using Abstraction and Regular Languages[C]// International Conference on Concurrency Theory. Springer-Verlag, 1995:395-407.
- [26] Bae K, Meseguer J. Model checking linear temporal logic of rewriting formulas under localized fairness[J]. Science of Computer Programming, 2015, 99:193-234.
- [27] Cavada R, Cimatti A, Dorigatti M, et al. The nuXmv Symbolic Model Checker[C]// International Conference on Computer Aided Verification. Springer-Verlag New York, Inc. 2014:334-342.
- [28] Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge, 2001.
- [29] Bonnet F, Potop-Butucaru M, Tixeuil S. Asynchronous Gathering in Rings with 4 Robots[M]// Ad-hoc, Mobile, and Wireless Networks. Springer International Publishing, 2016.
- [30] D' Angelo G, Stefano G D, Navarra A, et al. Computing on Rings by Oblivious Robots: A Unified Approach for Different Tasks[J]. Algorithmica, 2015, 72(4):1055-1096.
- [31] Huth M, Ryan M D. Logic in computer science - modelling and reasoning about systems (2. ed.)[J]. Ceskoslovenská Oftalmologie, 2004, 29(4):295-7.
- [32] Kawamura A, Kobayashi Y. Fence Patrolling by Mobile Agents with Distinct Speeds[C]// International Symposium on Algorithms and Computation. Springer Berlin Heidelberg, 2012:598-608.

- [33] Suzuki I, Yamashita M. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns.[C]// Sirocco'96, the, International Colloquium on Structural Information Communication Complexity, Siena, Italy, June. DBLP, 1999:313-330.
- [34] Clerentin A, Delahoche L, Brassart E, et al. Imprecision and Uncertainty Quantification for the Problem of Mobile Robot Localization[J]. Autonomous Robots, 2008, 24(3):267-283.
- [35] D'Angelo G, Stefano G D, Navarra A. Gathering of Six Robots on Anonymous Symmetric Rings[M]// Structural Information and Communication Complexity. Springer Berlin Heidelberg, 2011:174-185.
- [36] Kamei S, Lamani A, Ooshita F, et al. Gathering an Even Number of Robots in an Odd Ring without Global Multiplicity Detection[M]// Mathematical Foundations of Computer Science 2012. Springer Berlin Heidelberg, 2012:542-553.
- [37] Flocchini P, Prencipe G, Santoro N, et al. Gathering of asynchronous robots with limited visibility  $\square$ [J]. Theoretical Computer Science, 2005, 337(1):147-168.
- [38] Millet L, Potop-Butucaru M, Sznajder N, et al. On the Synthesis of Mobile Robots Algorithms: The Case of Ring Gathering[J]. Computer Science, 2014, 8756:237-251.
- [39] D'Angelo G, Stefano G D, Navarra A. How to gather asynchronous oblivious robots on anonymous rings[C]// International Conference on Distributed Computing. Springer-Verlag, 2012:326-340.
- [40] D'Angelo G, Stefano G D, Navarra A. Gathering on rings under the Look-Compute-Move model[J]. Distributed Computing, 2014, 27(4):255-285.
- [41] D'Angelo G, Stefano G D, Navarra A. Gathering six oblivious robots on anonymous symmetric rings  $\square$ ,  $\square\square$ [J]. Journal of Discrete Algorithms, 2014, 26:16-27.

- [42] D' Angelo G, Navarra A, Nisse N. Gathering and Exclusive Searching on Rings under Minimal Assumptions[C]// International Conference on Distributed Computing and Networking. Springer Berlin Heidelberg, 2014:149-164.
- [43] Stefano G D, Montanari P, Navarra A. About Ungatherability of Oblivious and Asynchronous Robots on Anonymous Rings[M]// Combinatorial Algorithms. Springer International Publishing, 2015.
- [44] Izumi T, Izumi T, Kamei S, et al. Time-Optimal Gathering Algorithm of Mobile Robots with Local Weak Multiplicity Detection in Rings[J]. Ieice Transactions on Fundamentals of Electronics Communications Computer Sciences, 2013, E96.A(6):1072-1080.
- [45] Kamei S, Lamani A, Ooshita F, et al. Asynchronous Mobile Robot Gathering from Symmetric Configurations without Global Multiplicity Detection[J]. 2011, 6796:150-161.
- [46] Klasing R, Kosowski A, Navarra A. Taking Advantage of Symmetries: Gathering of many Asynchronous Oblivious Robots on a Ring[C]// Principles of Distributed Systems, International Conference, Opodis 2008, Luxor, Egypt, December 15-18, 2008. Proceedings. DBLP, 2010:446-462.
- [47] Mostéfaoui A, Rajsbaum S, Raynal M, et al. Condition-based consensus solvability: a hierarchy of conditions and efficient protocols[J]. Distributed Computing, 2004, 17(1):1-20.
- [48] Ooshita F, Tixeuil S. On the self-stabilization of mobile oblivious robots in uniform rings □[J]. Theoretical Computer Science, 2015, 568(C):84-96.

## 攻读硕士学位期间发表论文和科研情况

### ■ 已公开发表论文