2018 届研究生硕士学位论文 学校代号: 10269
学号: 51151500019

華東師範大學

基于协同仿真和统计模型检测的信息物理融合系统验证分析方法

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: 可信软件
指 导 教 师: 杜德慧 副教授
硕士研究生: 姜 凯强

2018 年 5 月

2016 MASTER'S DISSERTATION School Code: 10269
Student Number: 51131500003

EAST CHINA NORMAL UNIVERSITY

Statistical Model Checking Based on Abstraction and Learning

Department:	School of Computer Science and Software Engineering
Major:	Software Engineering
Research Direction:	Trustworthy Software
Supervisor:	Associate Professor Dehui Du
Candidate:	Bei Cheng

May, 2016

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《基于抽象和学习的统计模型检测研究》，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

《基于抽象和学习的统计模型检测研究》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）。

程贝 硕士学位论文答辩委员会成员名单

姓名	职称	单位	备注
缪淮扣	教授	上海大学	主席
胡豪东	高级工程师	中航工业航空动力控制系统研究所	
陈铭松	教授	华东师范大学	

摘　要

信息物理融合系统（Cyber Physical Systems，CPS）是一种更关注计算机与物理环境交互和协作的高级嵌入式系统，自 2006 年此概念被提出以来，已受到了学术界与工业界的高度关注。第一，CPS 应用大都安全攸关或功耗要求严苛，在保证功能的前提下，仍必须满足一定的非功能属性，如吞吐量、能耗等，因此需要验证分析以保证其可信性；第二，CPS 大都是异构的混成系统，融合了连续的物理过程和离散的系统行为，且处于高度不确定的开放环境中，因此使用传统的方法（如模型检测和定理证明）难以完成验证分析。为缓解此问题，人们开始尝试使用统计算法对系统模型的仿真 Trace 进行分析，求得近似结果，并给出结果的误差范围，这种方法被称为统计模型检测（Statistical Model Checking，SMC）。SMC 无需遍历状态空间，但当结果精度要求较高时需要产生大量 Trace（多数仿真软件的 Trace 生成比较耗时），性能因此大大降低，本文即针对 SMC 的性能问题展开深入研究。

首先，对已有 SMC 算法的原理进行了剖析，实现了 4 种 SMC 算法，通过大量实验给出了详细的性能评估。基于实验结论，提出了一个自适应的 SMC 算法框架，以根据不同属性的预估概率，动态地选择合适的 SMC 算法。

其次，为改进自适应的 SMC 中贝叶斯区间估计算法的不足，提出了基于抽象和学习的 SMC 方法，旨在减少统计分析所需的 Trace 数量以提高 SMC 的效率。其中结合已有的抽象和学习理论（如主成分分析、随机文法推断），对随机混成自动机的仿真 Trace 进行了概率等价抽象和简化；并基于抽象 Trace 学习出概率等价的系统行为模型——前缀频率树，同时提出了树的两阶段约减算法，以有效控制树的规模，为更高效的 SMC 验证分析提供了良好的抽象模型。

最后，介绍了我们实现的 CPS 建模分析平台——Modana，基于此平台实现了本文提出的 SMC 改进算法，基于 Modana 平台建模分析了典型的 CPS 系统——智能温控系统；并结合 3 个基准测试案例，对 SMC 算法改进前后的性能和准确度进行了实验性评估。结果证明，本文提出的 SMC 改进方法正确并且有效。

关键词： 信息物理融合系统；随机混成自动机；主成分分析；统计抽象；统计模型检测

ABSTRACT

Cyber Physical Systems (CPS) are advanced embedded systems concerning more the interaction and collaboration between computer and physical environment. Since 2006 when this concept was presented, they have been highlighted by both academic and industrial worlds. First, most CPS applications are safety-critical or limit demanding energy consumption; a number of non-functional requirements (e.g. throughput, energy consumption, etc) need to be met when functional ones have been guaranteed, so that they required to be checked to achieve trustworthy systems. Second, most CPS are heterogeneous hybrid systems which combine continuous physical process and discrete system behavior, and also are exposed to open environment of high uncertainty; so traditional methods (model checking and theorem proving) can hardly finish checking effectively. To mitigate this issue, statistical methods are used to analyze the traces drawn from system simulator, by which an approximate result are obtained with an error bound. This method is known as Statistical Model Checking (SMC) which does analysis without traversing the state space of systems. However, SMC with high precision usually consumes a large number of traces (generating traces is seriously time-consuming for most simulation softwares), which leads to poor performance. This paper intensely studies the performance issue of SMC.

First of all, we gives an insight into the theory of existing SMC algorithms and implement four of them for conducting large numbers of experiments of their performance in detail. Based on our conclusion, an adaptive SMC algorithm framework is presented to automatically choose appropriate SMC algorithms according to the estimated proba-

bility of properties in different cases.

Next, to overcome the shortcoming of Bayesian Interval Estimate in the adaptive SMC, we present an SMC method based on abstraction and learning, aimed at improving the efficiency of SMC via reducing the number of traces for statistical analysis. This method uses the existing related learning theories (e.g. principal components analysis and stochastic grammar inference) to abstract and simplify probabilistically equivalent traces of Stochastic Hybrid Automata. Then we learn the probabilistically equivalent system behavior model, i.e. prefix frequency tree with abstracted traces, and effectively control the size of the tree by two-phase reduction algorithms presented also in this paper. It provides a well abstract model for more efficient verification and analysis with SMC later.

Finally, we introduce Modana Platform for modeling and analysis of CPS, which is implemented by our team. Based on Modana, we further implements the improved SMC presented in this paper. And a typical CPS application - smart heating system is modeled and analyzed. Then we experimentally evaluate the performance and accuracy of both original and improved SMC algorithms with three benchmarks. It turns out that our method is correct and efficient.

Keywords: *Cyber physical systems, Stochastic hybrid automata, Principal components analysis, Statistical abstraction, Statistical model checking*

目录

第一章　绪 论

1.1　研究背景及意义

信息物理融合系统（Cyber Physical System，CPS）[1] 是一种复杂的异构系统，主要具有以下两个特征：1) 不确定性：该系统面向开放环境，受多种不确定因素影响，如天气突变、信号误差、人为失误等，因此在设计该系统时，我们要充分考虑多种不确定性的因素，以保障 CPS 在不确定环境下的可信性；2) 异构性：该系统除了包含计算机系统之外，还结合了机械、环境、土木、电子、生物、化学、航空等诸多工程领域的模型和方法，且各领域深度的融合，因此我们不仅仅需要设计和建模各个工程领域的模型，更要充分考虑各个领域之间的交互问题。因此，由于 CPS 的不确定性和异构性，使 CPS 的建模分析及验证面临巨大的挑战，已有的 CPS 研究已经针对 CPS 的建模、仿真做了大量的工作并相应的有了一些工具的支持，例如基于时间自动机理论的 UPPAAL[2] 可以很好的建模和验证系统的基于时间的行为，基于马尔科夫模型的 Prism[3] 对概率系统的建模和验证提供了很好的支持，Modelica [4]、Simulink[5] 等工具支持对连续的物理系统提供较好的建模仿真等等。然而，以上工作在解决 CPS 的建模仿真及验证问题上各有优势及不足。因此，想要更好的支持 CPS 系统各种性质的建模、仿真和验证，我们需要整合多种特定领域模型工具的建模、仿真及验证优势。为了解决这一问题，GEORG, H 等人在论文 [6] 中提出了两种方法：一种是设计统一的建模仿真平台，该平台可以结合所有领域模型的优势并支持所有领域模型的建模仿真；另一种是使用协同仿真 [7] 技术，仍然是在各个领域模型的建模工具中进行建模，然后将各个领域建模工具建模出的模型根据统一标准转化为一种标准模型，最后根据这一标准提供的接

口在仿真过程对模型进行仿真。Georg, H 等人在文中也提到，第一种方案实现起来比较困难，所以建议使用第二种方案。经过协同仿真过程之后，我们可以得到不同领域模型融合在一起仿真的迹（trace），由于仿真迹的可读性不高，通过对仿真迹进行直接观察我们很难评估分析系统的行为。因此，LEGAY A 等人提出了统计模型检测方法（Statistical Model Checking，SMC）[8]，该方法以模型的仿真迹和验证属性（Property）作为输入，最终将得到模型满足这一属性的概率值。统计模型检测方法提出以来，主要是用来对特定领域模型的行为进行定量的评估分析，在本文中我们用该方法来评估分析多个模型协同之后的行为。

本文提出了一种基于协同仿真及统计模型检测的信息物理融合系统定量评估分析方法，在该方法中我们使用协同仿真技术来对多个领域模型进行融合仿真并得到仿真的迹，然后对整个系统设计需要验证的属性，最后将仿真迹和验证属性输入到统计模型检测器中进行评估分析。对于该方法，我们主要需要解决两个问题，也是本文的两个重要的贡献点：（1）需要保证多个领域模型进行协同仿真时的正确性，即需要对多个模型进行协同时的协同行为进行验证；（2）由于统计模型检测需要消耗大量的仿真迹，且在协同仿真时产生仿真迹的过程尤为耗时，使得整个验证时间需要极大的时间开销，因此我们需要对统计模型检测算法进行改进，以更快的完成验证过程。解决了以上两个问题之后，我们本文提出的方法就可以很好的结合多种建模仿真的优势，并为准确高效的对 CPS 系统的建模、仿真和定量评估分析提供有效的支持。

1.2 国内外研究现状

1.2.1 信息物理融合系统

信息物理融合系统的概念最早是由美国自然基金委提出，之后便获得了国内外的广泛关注。各个国家的科研人员从 CPS 的建模、验证分析等不同方面展开了深入的研究。CPS 从 2005 年提出至今，它的发展得到了多国政府的大力资助和支持，并成为学术界、科技界研究的重点方向，具有很高的科学研究意义。同时，CPS

也在工业界有了大量的案例应用，具有广阔的应用前景和商业价值。在美国，近些年举办了多次 CPS 的科研会议和研讨活动，针对 CPS 的理论、性能及安全性等问题展开了深入的探讨。美国科研人员的研究热点主要涉及嵌入式系统、网络信息安全等方面，并已经取得了较好的研究成果。例如：哥伦比亚大学伯克利分校设计的 PTOLEMY 平台对 CPS 系统的架构建模和仿真提供了工具支持。麻省理工学院（Massachusetts Institute of Technology, MIT）设计了分布式智能机器人花园，提高了 CPS 各个节点之间的交互和实时通讯的效率。宾夕法尼亚工程学院了汽车导航软件 GrooveNet，为车辆 CPS 系统的建模和仿真提供了一个良好的建模和仿真平台。Carnegie Mellon 大学将支持向量机预测模型和马尔科夫状态控制等方法应用于智能电网 CPS 系统的建模之中，来实现智能电网的调度控制。在欧洲，对于 CPS 的研究主要集中在理论研究方面。例如：在 2008 年，欧盟启动了 ARTEMIS (Advanced research and technology for embedded intelligence and systems) 项目, 将 CPS 作为一个重点的研究方向, 并创办了“International Journal of Cyber-Physical Systems”专刊。法国自动化研究所设计的 GEMOC 平台对信息物理融合系统的建模提供了有效的支持。在中国，于 2008 年在北京召开了 IEEE 嵌入式研讨会, 将 CPS 作为接下来的重点研究方向。2010 年, 国家 863 计划信息技术领域办公室和专家组在上海举办了“CPS 发展战略论坛”, 对 CPS 给以高度关注。武汉大学信息资源研究中心提出了结合云计算和下一代互联网的理念, 进行 CPS 语义中间件的设计, 研了究 CPS 网络互联和自主交互等技术。华东师范大学使用形式化的方法对 CPS 进行了建模，并对其可信性进行了验证分析。此外, 清华大学、天津大学、同济大学等多所研究机构也对 CPS 进行了深入研究。

1.2.2 协同仿真

CPS 包含信息和物理两个部分，并涉及各个领域。因此，对于 CPS 的各个属于不同领域的模块都有相应的工具及方法支持。对 CPS 的建模和仿真主要有两种方案: （1）开发一个统一的 CPS 建模平台，该平台支持 CPS 涉及的各个领域的建模和仿真（2）将 CPS 的各个部分在不同的领域工具中建模，使用协同仿真技术在

仿真阶段将不同领域模型融合在一起进行仿真。方法一到目前为止实现起来较为复杂，因此学术界对协同仿真技术进入了深入的研究。为了模拟 CPS，H. Georg 等人在论文 [6] 中提出整合 CPS 的不同仿真域来综合分析 CPS 中相互依赖的子系统。作为一种十分有效的 CPS 仿真技术，协同仿真可以将让 CPS 不同的领域模型在它们自己的专有仿真软件之中进行仿真并在特定的时间点进行模型之间的数据交互以到达模型的数据同步。功能模拟接口（Functional Mock-up Interface，FMI）[9] 是一个行业标准，它能够使用多个仿真引擎对复杂异构系统进行协同仿真，已被业界和学术界所采用。例如，J. Bastian 等人在论文 [7] 中采用 FMI 标准和固定步长主算法来对异构系统进行协同仿真。实现了 FMI 标准的组件被称为功能模拟单元（Functional Mock-up Unit，FMU）[10]，D.Broman 等人在论文 [11] 中讨论了在协同仿真过程中，异构系统中各个 FMU 进行组合时的确定性问题。此外，他们对 FMI 标准进行了扩展并提出了支持回滚和步长预测主算法，以提高协同仿真的准确度和效率。在论文 [12] 中，F. Cremona 等人介绍了基于 FMI 标准构建的协同仿真集成开发环境 FIDE。在我们之前的研究工作 [13] 中，我们基于 FMI 标准实现了 prism 和 modelica 模型的协同仿真，并且提出了一种基于事件的协同仿真算法，一定程度上提高了协同仿真的效率。由于协同仿真存在不确定性，因此在进行仿真之前我们应该保证整个系统的协同行为的正确性。针对协同行为正确性的验证，相关研究人员也做了一定的研究工作。例如：P.G. Larsen 等人在论文 [14] 中使用形式化规约语言 CSP 对 FMI 的语义进行了形式化的描述，并使用验证工具 FDR3 对得到的模型进行了形式化的验证分析。

1.2.3 分布式技术

分布式技术与传统的集中式的处理技术相对应，它是一种基于网络的多个计算机并行计算的处理技术。随着信息技术的发展，系统变得愈加复杂，因此对计算机的处理能力有了更高的要求。在一定程度上，单台计算机的处理能力已经很难满足当前复杂系统的计算需要。分布式技术使用多台低成本的计算机达到了十分高的计算性能。该技术近些年来发展十分迅速，并且在大数据存储、高复杂度的数

据计算等方面发挥了重要作用。当前分布式技术引起了学术和工业界的高度重视，并且已经在科研、医疗、教育等方面有了许多应用案例。例如：SETI@home 通过使用参与分布式的计算集群来对射电望远镜收到的海量信号进行计算分析，以确定还有无类似地球生物的生命存在，寻找宇宙生命体。Climateprediction 工程是一个分布式计算技术在气象预测领域的一个成功案例，该项目将收集到的海量气象数据发放给多台计算机，每台计算机都会对接收到的数据继续计算并把计算结果返回，最终组建了地球的气象模型。Folding@home 是分布式计算技术在医疗领域的一个成功的应用案例，这个项目基于多个计算机组成的超高计算性能来研究蛋白质折叠、聚合及由此引起的疾病。

1.2.4 统计模型检测

统计模型检测概念基本算法在论文 [15] 中，R.Grosu 等人首次提出了统计模型检测的概念。统计模型检测基于仿真和统计技术，来验证某个模型满足某一验证属性的概率或者是验证某个模型满足某一属性的概率是否大于或小于某个特定的阈值。根据最终验证目标的不同，可以将统计模型检测算法分为定性和定量算法。定性算法回答了“系统满足属性的概率是否大于或等于某个特定阈值”的问题，最终得到的结果是一个布尔值。当前经典的定性统计模型检测算法主要有 Single Sampling Plan（SSP）[16] 算法、Sequential Probability Ratio Test（SPRT）[17] 和 Bayesian Hypothesis Testing（BHT）[5] 算法。SSP 算法最早被 Sen 等人提出，但是该算法难以确定算法收敛需要的样本数量和接受假设的阈值，Younes 等人在论文 [18] 中提出使用二叉搜索的算法来近似得到这两个参数值。在之后，Younes 等人在论文 [18, 19] 中还提出了 SPRT 算法的设计和算法实现，用来提高定性统计模型检测算法的效率。Zuliani 等人也基于贝叶斯统计理论提出了 BHT 算法来提高定性统计模型检测算法的效率。定量算法用来直接得到模型满足某一验证属性的概率，当前经典的定量统计模型检测算法主要有 Approximate Probabilistic Model Checking（APMC）[20] 算法和 Bayesian Interval Estimation（BIE）[21] 算法，APMC 算法用满足属性的样本数量除以所有得到的样本数量来得到满足验证属性的概率，并用

Chernoff-Hoeffding 界来限定误差。Clarke 等人基于贝叶斯统计理论提出了 BIE 算法以提高定量统计模型检测的效率。Kim 等人在论文 [22] 给中出了以上算法的对比分析，实验结果表明以上几种算法均可以对模型进行有效的定量或定性的评估分析。对于定性统计模型检测算法，SPRT 和 BHT 算法比 SSP 算法更加高效，且当概率阈值远离最终的真实概率时，BHT 比 SPRT 高效，否则 BHT 的效率明显低于 SPRT。对于定量的统计模型检测算法，Zuliani 等人在论文 [5] 中对比了它们使用的样本数量并得出结论，BIE 算法比 APMC 算法更加高效。

由于统计模型检测算法需要消耗大量的仿真样本，其验证过程较为耗时，因此其效率问题受到了广泛的关注。相关研究人员提出了使用抽象和学习的方法来提高统计模型检测的效率。例如：在论文 [23] 中，Kumar 等人通过预测可能的结果或事件的发生来学习预测模型并进行验证，这种方法可以直接应用现有的机器学习技术，但很难得到精确的误差评估。在论文 [24] 中，等人使用在概率等价的基础上，使用抽象的方法将原始的模型进行简化，然后对简化之后的模型进行验证，该方法在保证准确度的基础上一定程度提高了统计模型检测的效率。在我们之前的工作 [25] 中，我们使用抽象和学习的方法将原始模型划分成了多个子模型，然后对多个子模型进行验证分析，最终将验证结果合并。该方法可以有效提高统计模型检测的效率并将误差控制在了一定的范围之内。在论文 [26] 中，Younes 等人指出统计模型检测算法可以进行分布式设计，并且他们还提出了一种方法来消除分布式设计中可能出现的偏差。接下来，Peter Bulychev 等人在论文 [27] 中对 BHT 算法给出了分布式的设计和实现。

当前，已经有越来越多的工具支持了统计模型检测算法。Ymer[26] 和 Vesta[28] 最早对统计模型检测算法提供了支持。Vesta 采用了 SSP 算法的一个变种，而 Ymer 采用的是 SPRT 算法，且 Ymer 工具的效率要高于 Vesta。目前对 SMC 算法提供较好支持的工具是 UPPAAL-SMC 和 Prism，UPPAAL-SMC 和 Prism 都实现了定性的 SPRT 算法和定性的 BIE 算法，同时它们还实现了都实现了 Confidence Interval（CI）[29] 算法。UPPAAL-SMC 时间自动机理论，对时间和连续性有较好的支持且有图形

化的建模界面；Prism 基于马尔科夫模型，对随机性和不确定性有较好的支持，且使用 Reactive Modules Language（RML）建模。Plasma Lab[30] 是一个统计模型检测算法的集成工具，其中实现了 BHT,BIE,SPRT 等多种统计模型检测算法。用户可以根据需要选择要用的统计模型检测算法并输入参数进行模型的评估分析，该工具同样使用 RML 进行建模。

1.3 本文技术路线及主要研究内容

信息物理融合系统是异构系统，本文针对这种异构系统的验证提出了一种解决方案，图1.1为本文的技术路线图，本文的技术路线大致如下：

（1）我们通过对该系统进行分析，提取出该系统的信息部分（Cyber part）和物理部分（Physical part），除此之外我们根据自己需要验证的行为属性定义约束（Constraint）。

（2）使用系统建模语言（(Systems Modeling Language，SysML）[31] 建模语言对提取出的信息部分和物理部分进行建模，将信息系统和物理系统的组件使用 SysML 的块定义图（SysML Block Definition Diagram，BDD）图进行建模，同时使用 SysML 的内部框图（SysML Internal Block Diagram，IBD）图来描述系统中各个组件之间的关联。

（3）SysML 只是用来建模组件内部结构和组件之间的关联，该模型不可直接进行仿真运行，因此我们将 SysML BDD 图建模的模型使用 FMU 进行实现，同时将 SysML IBD 图描述的系统组件关系转化为 FMU 之间相互依赖的接口配置文件，此时，我们只需要再设计好协同仿真的主算法（Master Algorithm）就可以进行异构系统的协同仿真。然而，在进行协同仿真之前，我们首先要保证各个 FMU 之间的协同是正确的，要确保 FMU 之间协同行为的正确性，我们需要验证主算法的正确性及各个 FMU 之间的连接顺序及数据交换的正确性。在本文中，我们基于时间自动机设计了一个协同行为正确性验证的验证器，我们将系统的多个 FMU、协同仿真的主算法以及 FMU 之间的接口配置文件输入到该协同行为的验证器之中即可验

证当前系统协同行为的正确性，如果验证通过则说明我们当前的模型即为正确的模型，如果验证不通过，则需要修改当前系统的协同行为，直到得到验证通过的模型之后再输入到仿真器中进行仿真。

（4）我们在进行系统的验证分析时，首先需要验证的系统模型，同时我们还需要验证的属性（Property），我们将（1）中得到的约束进行形式化描述，即可得到验证属性（该验证属性根据约束的不同可以是 BLTL/ALTL/GSCL 等等，本文主要使用了 BLTL 作为验证属性）。

（5）将通过第三步验证的模型（Verified Model）及第四步得到的验证属性输入到异构系统验证器（co-verification）之中进行验证分析，首先将模型输入到仿真器（Simulator）之中进行仿真或协同仿真（Co-simulation）并得到仿真迹（Traces），然后将得到的仿真迹和验证属性输入到模型验证器（Checker）之中来验证该迹是否满足某条特定的验证属性，得到结果满足为 1，不满足为 0，我们将该验证是否满足的结果称为观察值（Observations），多条仿真迹对于一条特定的验证属性会得到多个观察值。最后将得到的观察值输入到统计分析算法中进行统计分析，并得到评估结果。

本文的具体研究内容和贡献点总结如下：

1. 使用 SysML 建模语言建模整个系统的架构，使用 SysML 的 BDD 图建模系统组件，使用 SysML 的 IBD 图来描述系统中各个组件的关联关系。

2. 基于 FMI 标准实现 SysML 描述的系统模型，将 SysML 的 BDD 图建模的系统组件包装成 FMU，并将各个组件的关联关系转化为 FMU 之间的接口配置文件。

3. 使用时间自动机理论验证了基于 FMI 标准的多个 FMU 的协同行为，使用时间自动机将协同仿真的主算法进行形式化描述，并使用 UPPAAL 模型检测器来验证主算法的正确性；提出了一种从 FMU 到时间自动机的映射标准，通过此标准用时间自动机将多个 FMU 进行编码，并用时间自动机之间的通道

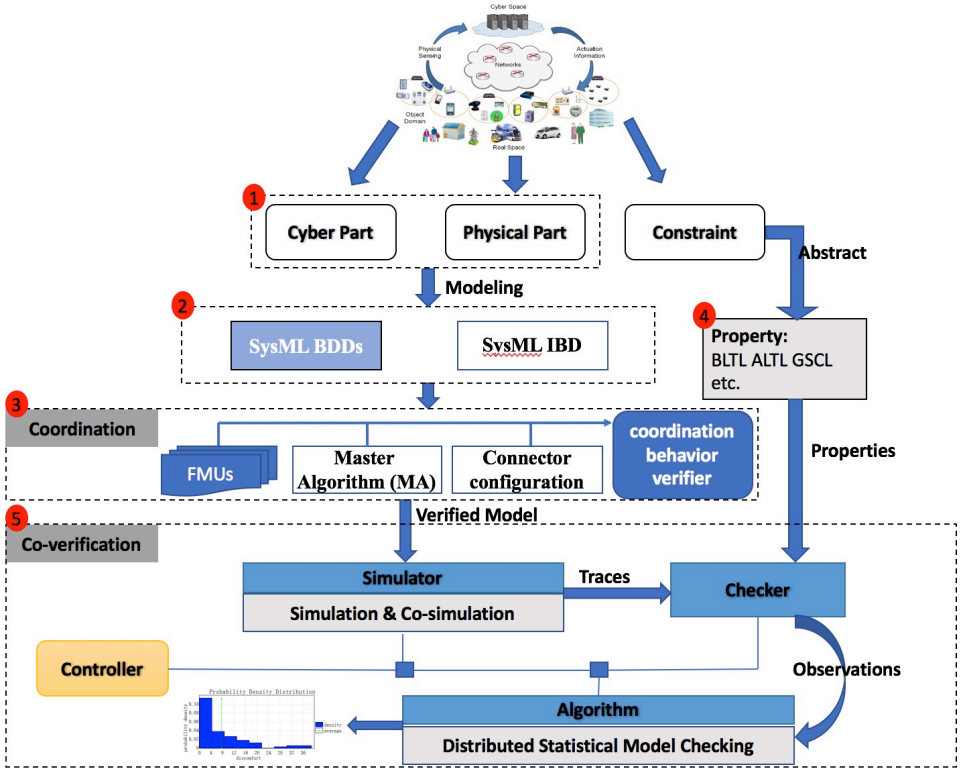


图 1.1: 论文技术路线图

（channel）来描述多个 FMU 之间的关联关系，最终将多个 FMU 及 FMU 之间的关联关系使用一个时间自动机网络进行了形式化描述，将该时间自动机网络输入到 UPPAAL 之中进行验证从而来验证协同行为的正确性。

4. 提出了一种基于抽象和学习的分布式统计模型检测算法，在将误差控制在有效范围之内的基础上大大提高了统计模型检测的效率。

1.4 本文组织结构

本文共分七章，组织结构如下：

第一章介绍了本文的研究背景及意义，并从四个方面阐述了该研究领域的国内外研究现状，其中包括信息物理融合系统的形式化建模、分布式技术、协同仿真及统计模型检测的研究现状。之后，给出了本文的技术路线和主要贡献点。最后，总结了论文组织结构。

第二章介绍了相关预备知识。首先给出了 FMI 标准的主要概念及定义，之后给出了概率有界线性时态逻辑、时间自动机及 FMU 的语法和语义定义。

第三章介绍了基于时间自动机理论来验证异构系统协同行为正确性的方法，首先我们给出了该方法的技术框架，之后我们详细描述了如何用时间自动机理论来验证协同仿真的主算法，以及如何验证整个异构系统的协同行为的正确性。

第四章重点阐述了如何用统计模型检测算法来对异构系统进行验证分析，也是本文的主要内容。首先，介绍了如何基于 FMI 标准对异构协同进行协同仿真并得到仿真迹，然后提出了一种基于抽象和学习的统计模型检测算法来提高统计模型检测的效率。最终，我们将协同仿真和该高效的统计模型检测算法进行结合，以此来对异构系统进行验证分析。

第五章主要介绍工具及程序实现。首先简单介绍了我们设计的异构系统验证工具（Co-SMC 工具）。之后，给出了 Co-SMC 工具的详细设计及程序实现。

第六章给出了两个案例，通过使用本文提出的方法对这两个案例进行建模、仿真和分析来验证本文提出方法的有效性。

第七章为总结和展望，总结了本文提出的基于协同仿真和统计模型检测的信息物理融合系统验证方法，并讨论了其优点和不足，指出了未来要进一步进行研究的工作。

1.5 本章小结

本章首先说明了选题的背景和意义，指出了由于 CPS 的异构性而导致 CPS 系统的验证分析面临巨大挑战；接着介绍了信息物理融合系统、分布式技术、协同仿真及统计模型检测的研究现状；最后给出了本文的技术路线、主要贡献点和组织结构。下一章将介绍本文涉及到的预备知识及概念。

第二章　预备知识与概念

2.1　功能模拟接口（FMI）

CPS 中各个组件之间的协同仿真可以基于 FMI 标准来实现，FMI 标准最初是在 2008 年开始的 MODELISAR 项目中开发的，并得到大量软件公司和研究中心的支持。FMI 支持模拟由异构组件组成的复杂系统，通过一个协同仿真环境将不同模型与自己的求解器耦合起来。实现了 FMI 标准接口的系统组件被称为 FMU，我们可以基于 FMI 标准将模型转化为 FMU，之后加入协同仿真的主算法就可以对系统进行协同仿真，其中主算法并不是 FMI 标准的一部分。FMI 标准包含两个主要部分：

1. 协同仿真接口：一组用来实现对仿真器控制和完成多个 FMU 之间进行数据传输的 C 语言函数。

2. 协同仿真描述文件：用一个 XML 文件定义了模型的结构和主要的描述信息。主要包括模型的输入、输出信息，模型的仿真器和求解器等等。

FMI 至今已经有两个版本，即 FMI1.0 和 FMI2.0[11]。FMI1.0 中有一个主要的函数是 doStep，主函数可以调用该函数，使得 FMI 中的某个 FMU 进行一步的仿真。该函数的定义如下：

fmiStatus fmiDoStep(fmiComponent c, fmiReal currentCommunicationPoint, fmiReal communicationStepSize, fmiBoolean newStep);

其中，c 为主算法要调用的 FMU 组件，$currentCommunicationPoint$ 为主算法当前的仿真时间，$communicationStepSize$ 是主算法要 FMU 下一步进行仿真的步长。FMU 接收到主算法的调用命令和调用步长时，将会接受或拒绝这一步长，

如果接受这一步长则到进行仿真并到达一个新的状态，如果拒绝则需要重新调用。$newStep$ 用来表示当前的这一次仿真是一次新的调用还是拒绝之后的再次调用。

FMI2.0 针对 FMU1.0 做了一定的扩展，主要是增加了 $fmiGetFMUstate$ 和 $fmiSetFMUstate$ 两个方法。这两个方法使得 FMU 可以对状态进行保存和恢复，从而支持了 FMU 状态的回滚。在后续章节，我们提出的支持回滚的主算法是基于这一扩展进行设计的。因此，扩展之后的 doStep 函数如下：

fmiStatus fmiDoStep(fmiComponent c, fmiReal currentCommunicationPoint, fmiReal communicationStepSize, fmiBoolean noSetFMUStatePriorToCurrentPoint);

其中，前三个参数跟 FMU1.0 里面的参数相同，只是最后一个参数的含义与 FMI1.0 有了本质的区别，通过设置 $noSetFMUStatePriorToCurrentPoint$ 为 true 和 false，主算法可以来控制 FMU 是否将之前一个状态的值设置给当前时间节点。

2.2　功能模拟单元（FMU）

我们在以上小节介绍了 FMI 标准接口，下面我们对实现了 FMI 标准接口的 FMU 给出形式化的语法和语义。

定义 2.2.1. ***FMU* 语法**

FMU 的语法可以用一个八元组 $F = (S, U, Y, D, s_0, set, get, doStep)$ *表示，*

- S 表示 FMU 的状态集合。
- U 表示 FMU 的输入变量集合。
- Y 表示 FMU 的输出变量集合。
- $D \subseteq U \times Y$ 表示多个 FMU 之间输入和输出之间的依赖关系集合。$(u, y) \in D$ 表示输出变量 y 直接依赖于输入变量 u 的取值。
- $s_0 \in S$ 表示 FMU 的初始状态。

- $set : S \times U \times \mathbb{V} \to S$ 表示 set 函数对一个输入变量进行赋值。给定当前状态 $s \in S$, 输入变量 $u \in U$ 及一个数值 $v \in \mathbb{V}$, 该函数将返回一个新的状态，此状态中 u 的值为 v。

- $get : S \times Y \to \mathbb{V}$ 表示 get 函数返回某个输出变量的数值。给定一个状态 $s \in S$ 和一个输出变量 $y \in Y$, $get(s, y)$ 返回 s 状态上 y 输出变量的取值。

- $doStep : S \times \mathbb{R}_{\geqslant 0} \to S \times \mathbb{R}_{\geqslant 0}$ 表示 $doStep$ 函数进行了一步仿真。给定当前状态 s 和一个非负实数 $h \in \mathbb{R}_{\geqslant 0}$，$doStep(s, h)$ 返回一个元组 (s', h')，且
  当 $h' = h$ 时, F 接受该步长并执行步长为 h 的仿真，并且迁移到了一个新的状态 s';
  当 $0 \leqslant h' < h$ 时，F 拒绝步长 h，只执行步长为 h' 的仿真，并且迁移到一个新的状态 s'。

定义 2.2.2. *FMU* 语义

给定一个 FMU$F = (S, U, Y, D, s_0, set, get, doStep)$，FMU 的执行依赖于 $doStep$ 函数，它的执行可以用一个时间输入序列（Timed Input Sequence, TIS）进行描述。TIS 是一个有限的四元组序列 (t, s, v, v'), $t \in \mathbb{R}_{\geqslant 0}$ 表示当前时刻，$s \in S$ 表示 F 的当前状态，v 是一个输入变量，$v' : Y \to \mathbb{V}$ 是一个输出变量。

TIS = $(t_0, s_0, v_0, v'_0), (t_1, s_1, v_1, v'_1), (t_2, s_2, v_2, v'_2), ..., (t_i, s_i, v_i, v'_i), (t_{i+1}, s_{i+1}, v_{i+1}, v'_{i+1}), ...$
定义如下：

- $t_0 = 0$ 时刻的 s_0 状态表示 F 处于初始状态位置。

- 对于任意的 $i \geqslant 1, t_i = t_0 + \sum_{k=1}^{i} h_k$。

- 给定当前状态 s_i，set 函数用来将当前状态的输入参数设置为一个特定的数值 v，之后 F 执行 $doStep$ 函数并且迁移到一个新的状态 s'_i。get 函数用来得到当前状态的所有输出参数的取值 v'_i。

2.3　时间自动机（TA）

时间自动机 [?] 是一个建模实时系统行为的经典理论模型。在本小节中，我们给出时间自动机的形式化语法和语义。

定义 2.3.1. 时间自动机语法

时间自动机可以用一个四元组 $A = (L, l_0, E, I)$ 来表示，其中：

- L 表示时间自动机中有限的位置集合；
- $l_0 \in L$ 为时间自动机的初始位置；
- 约束集合 $G(x)$ 可以用 $g = x \bowtie c \mid g \wedge g$ 来表示，其中 $x \in X, c \in \mathbb{N}$ 且 $\bowtie \in \{<, \leqslant, \geqslant, >, =\}$。
- $E \subseteq L \times G(X) \times Act \times 2^X \times L$ 是一组边的集合，该边包含约束和时钟，其中 $Act = Act_i \cup Act_o$。Act_i 是一个输入动作的集合，Act_o 是一个输出动作的集合。
- $I : L \to G(X)$ 将不变量指定给位置。

定义 2.3.2. 时间自动机语义

时间自动机 $A = (L, l_0, E, I)$ 的语义可以用一个标签迁移系统 $L_A = (Proc, Lab, \{\xrightarrow{\alpha} \mid \alpha \in Lab\})$ 进行描述，其中：

- $Proc = \{(l, v) \mid (l, v) \in L \times (X \to \mathbb{R}_{\geqslant 0})$ 且 $v \models I(l)\}$，其中状态是一个 (l, v) 元组，l 是时间自动机中的位置且 v 是满足 $I(l)$ 的一个时钟变量；
- $Lab = Act \cup \mathbb{R}_{\geqslant 0}$ 是一个标签集合；
- 迁移关系定义如下：

  $(l, v) \xrightarrow{\alpha} (l', v')$，如果存在一个边 $(l \xrightarrow{g,\alpha,r} l') \in E$，则 $v \models g, v' = v[r]$ 且 $v' \models I(l')$

$(l,v) \xrightarrow{d} (l,v+d)$，对于所有的 $d \in \mathbb{R}_{\geqslant 0}$，则 $v \models I(l)$ 且 $v+d \models I(l)$

对于时间自动机 A 中某个位置 l 的可达性问题就是一个判断在迁移系统 L_A 中是否存在一个从初始状态 (l_0,v_0) 到状态 (l,v) 的路径。为了验证需要，我们定义了时间自动机的符号语义，该定义用到了一组包含时钟的执行序列集合。

对于一个特定的位置 l 和特定的时刻 $t \in X$，对于任意的 $x \in X$，则 $t+x \in X$。从该时刻位置开始的执行序列如下所示：

$(l,t) \xrightarrow{x_1} (l,t+x_1) \xrightarrow{x_2} (l,t+x_1+x_2) \xrightarrow{x_3} (l,t+x_1+x_2+x_3) \xrightarrow{x_4} ... \xrightarrow{x_i} (l,t+x_1+x_2+x_3+...+x_i) \xrightarrow{x_{i+1}} ...$

其中 $x_i > 0$ 且无穷序列 $x_1+x_2+...$ 对于 x 是收敛的。

2.4　系统建模语言（SysML）

SysML 是为模型驱动式软件开发（Model-Based Systems Engineering，MBSE）[31] 而提出的一种通用的领域建模语言（Domain-Specific Language，DSL）[?]，它起源于国际系统工程委员会（INCOSE）的倡议 cite Pepper2015International 并于 2001 年 1 月发布。SysML 基于统一建模语言 2.0（Unified Modeling Language2.0，UML2.0）[32] 的一个子集，加入了其他一些建模元素的扩张。SysML 的提出是用来统一复杂系统开发的各种建模语言和技术。SysML 是一种基于 UML 建模语言，针对系统工程做了相应的扩展而构造出的一种建模语言。它为系统的建模和需求建模提供了新的方法。在 [32] 中，Willard B. 重点介绍了 UML2.0 的发展，并提出了 SysML 带来的新的可能性，他声称 SysML 的主要优点是“为系统工程师提供标准和全面的系统规范范例”。SysML 使用 UML2.0 中的图作为类或对象图，但是采取了新的语义来避免软件词汇表（例如，类和对象被块代替）。此外，加入新的图来简化需求的声明，并针对之后的仿真提出了一些相关的设计。块（block）是 SysML 的基本概念。块是描述系统的每个模块单元的基本建模元素。块定义了一组代表组件结构和行为的特征和操作。这些块在 BDD 内以层级关系进行声明和组织。在 BDD 中，块拥有自己的属性，这些属性将它们的特征和子元素定义为部件和端口。

此外，块还拥有它们能够执行的一组操作。IBD 则显示了各个块的端口之间的连接关系。

2.5　概率有界线性时态逻辑（PBLTL）

概率有界线性时态逻辑（Probabilistic Bounded Linear Temporal Logic，PBLTL）[33] 公式可以用来形式化的描述系统的验证属性。我们先给出有界线性时态逻辑（Bounded Linear Temporal Logic，BLTL）的语法和语义，然后再将其扩展为 PBLTL。

给定一个模型 M，设其状态变量的集合 SV 是一个有限的实数集，在 SV 上的一个布尔谓词约束为 $y \sim v$ 形式。其中 $y \in SV$，$\sim \in \{\geq, \leq, =\}$ 且 $v \in \mathbb{R}$。BLTL 的语法定义如下：

$$\varphi ::= y \sim v \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi_1 \mid \phi_1 U^{\leq t} \phi_2$$

其中，$\sim \in \{\geq, \leq, =\}$，$y \in SV$，$v \in \mathbb{Q}$，$t \in \mathbb{Q}_{\geq 0}$。

定义算子“F”表示最终会满足，$F^{\leq t}\phi = TrueU^{\leq t}\phi$，表示最终在 t 时间内存在 ϕ 满足；

定义算子”G” 算子表示始终满足，$G^{\leq t}\phi = \neg F^{\leq t}\neg\phi$，表示在 t 时间内 ϕ 始终满足。

对于一条仿真迹 σ，σ^k 表示 σ 从第 k 步开始执行之后的部分。我们规定 $V(\sigma, k, y)$ 表示迹 σ 在第 k 步时状态变量 y 的值，t_k 表示第 k 步需要的时间，t 表示时间约束，则 BLTL 在迹 σ^k 上的语义定义如下：

定义 2.5.1. （有界线性时态逻辑的语义）.

- $\sigma^k \vDash y \sim v$ 当且仅当 $V(\sigma, i, y) \sim v$。
- $\sigma^k \vDash \phi_1 \vee \phi_2$ 当且仅当 $\sigma^k \vDash \phi_1$ 或 $\sigma^k \vDash \phi_2$。
- $\sigma^k \vDash \phi_1 \wedge \phi_2$ 当且仅当 $\sigma^k \vDash \phi_1$ 且 $\sigma^k \vDash \phi_2$。
- $\sigma^k \vDash \neg\phi_1$ 当且仅当 $\sigma^k \vDash \phi_1$ 不成立。

- $\sigma^k \vDash \phi_1 U^t \phi_2$ 当且仅当存在数值 i 使得

  $\Sigma_{0 \le l < i}\, t_{k+1} \le t$;

  $\sigma_{k+i} \models \phi_2$;

  $\sigma_{k+j} \models \phi_1$，对于每个 $0 \le j \le i$。

定义 2.5.2. （概率有界线性时态逻辑）.

一个 *PBLTL* 属性公式 ϕ 可以表示为 $P_{\ge\theta}(\phi')$，其中 ϕ' 为 *BLTL* 公式，θ 为一个阈值（介于 *0* 和 *1* 之间）。对于定性分析的 *PBLTL* 验证属性，我们将其表示为 $M \models P_{\ge\theta}(\phi)$，即来验证模型 M 满足 *BLTL* 属性 ϕ' 的概率是否不小于 θ。对于定量分析的 *PBLTL* 验证属性公式，无需指定阈值 θ，可以将其表示为 $M \models P_{=?}(\phi')$，即来验证模型 M 满足 *BLTL* 属性 ϕ' 的概率是多少。

2.6　本章小结

本章首先介绍了实现异构系统各个组件之间协同运行的标准，即 FMI 标准和 FMU 的语法及语义；同时，给出了用于验证组件之间协同行为正确性的理论模型-时间自动机的语法和语义。最后，给出了描述本文的验证属性主要用到的逻辑公式 PBLTL 的语法和语义。下一章我们将介绍如何建模整个异构系统的架构以及如何基于时间自动机理论来验证异构系统中各个组件协同行为的正确性。

第三章　异构系统协同行为的正确性验证

在本章节，我们首先用 SysML 建模语言对整个异构系统的架构进行建模，其中用 SysML 的 BDD 图来建模系统的组件，用 SysML 的 IBD 图来建模系统中各个组件之间的关联关系。由于 SysML 只是用来建模系统的架构，该建模语言得到的模型不可执行。因此，我们基于 FMI 标准，将 SysML 的 BDD 描述的组件用 FMU 进行实现，将 SysML 的 IBD 图转化为 FMU 之间的接口配置文件。因此，我们只需要设计协同仿真主算法就可以完成异构系统基于 FMI 的协同仿真。然而，在我们将基于 FMI 的可仿真模型输入到协同仿真引擎之中进行仿真之前，我们需要确保该模型的协同行为的正确性，即要保证该模型可以正确的进行协同仿真。为了解决这个问题，我们需要进行两个方面的验证:（1）我们要验证协同仿真主算法的正确性，即算法没有出现死锁以及其他一些特性；（2）我们要验证系统之间各个组件之间关联顺序及数据交互的正确性，其中最主要的要保证多个 FMU 没有出现环路依赖。由于 FMU 的执行时基于时间的，而时间自动机在建模时间有较大的优势，且时间自动机的验证有着良好的工具支持，因此，在接下来的内容中，我们重点描述如何基于时间自动机理论来验证整个可仿真模型的协同行为的正确性。在第一小节，我们给出整个验证过程的技术框架；第二小节介绍如何对上述的第一个方面进行验证，即协同仿真的主算法的验证；在第三小节中，我们使用一个具体的案例来描述如何对上述的第二个方面进行验证，即异构系统协同行为的验证。

3.1　技术框架

图3.1为异构系统协同行为正确性验证的技术框架图. 首先，我们在建模（Modeling）阶段使用 SysML 的 BDD 和 IBD 图来建模整个异构系统的架构。SysML 的

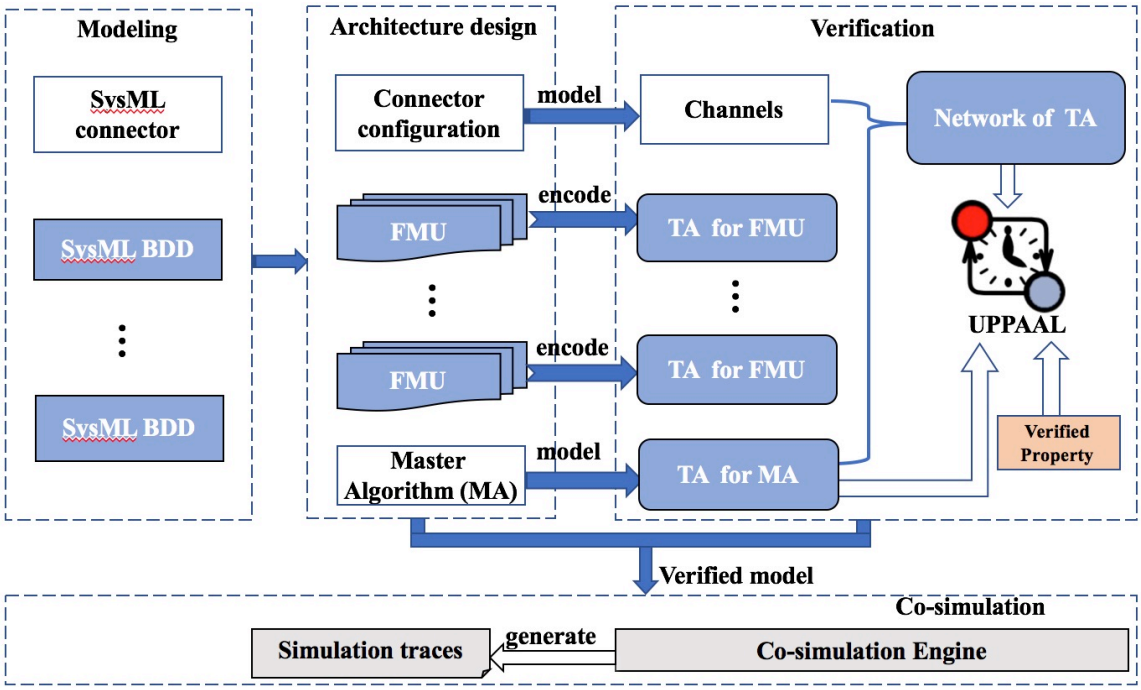


图 3.1: 协同行为正确性验证技术框架图

BDD 中的块建模了异构系统的一个个组件；SysML 的 IBD 图描述了各个块之间的连接关系。为了借助协同仿真技术对整个系统进行协同仿真，我们在架构设计（Architecture design）阶段将一个个块用 FMU 进行实现，同时将 IBD 图描述的关联关系转化为一个 FMU 之间的配置文件（Connector configuration）。接下来，我们自己设计了协同仿真的主算法（Master Algorithm，MA）来实现各个 FMU 之间的交互，同时来驱动协同仿真过程的执行。接下来是协同行为的验证（verification）阶段，也是我们本文的主要贡献点之一。为了验证协同行为的正确性，（1）我们先验证了协同仿真主算法的正确性：首先，我们用时间自动机对协同仿真的主算法进行形式化建模，然后将该时间自动机模型输入到 UPPAAL 模型检测工具中进行验证（2）我们验证了整个系统协同行为的正确性：我们提出了一种从 FMU 到时间自动机的映射规则，我们根据此规则将 FMU 映射为时间自动机，接下来将 FMU 之间的配置文件转化为时间自动机的通道（Channels）。这样，我们就得到了一个时间自动机网络（Network of TA）：包括 FMU 映射成的时间自动机、主算法的时间自动机及各个时间自动机之间的通道）。最后我们将该时间自动机网络和要验证的属

性（死锁、可达性、活性等）输入到 UPPAAL 中来验证该模型是否满足某些属性。一旦验证了协同行为的正确性，我们就可以将通过验证的模型输入到协同仿真的引擎之中进行协同仿真并得到协同仿真的迹。接下来，我们将详细介绍整个协同行为的验证过程。

3.2　协同仿真主算法的验证

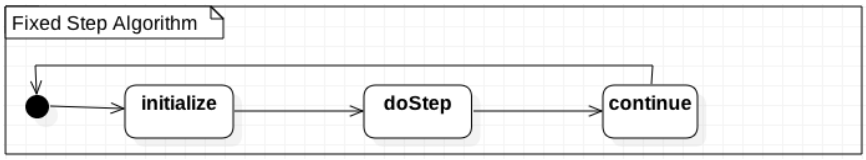
在本小节我们首先介绍了三种协同仿真的主算法：固定步长算法、可回滚算法及可预测步长算法，之后我们用时间自动机形式化建模了这几种算法，最后将这几种算法的形式化模型输入到 UPPAAL 工具中，分别验证了这几种算法的有无死锁、可达性和活性的属性。

3.2.1　协同仿真主算法介绍

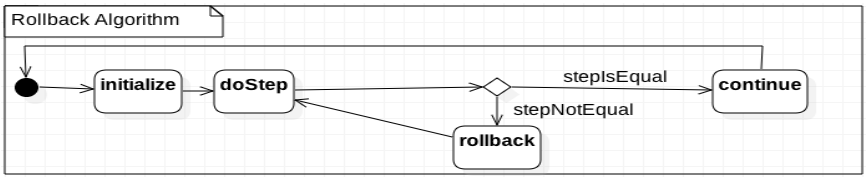
协同仿真的主算法用了调度和协同异构系统的多个 FMU 组件的执行。每一个 FMU 都可以被看做一个可独自仿真的黑盒，但是多个 FMU 之间在某些特定的时刻需要进行数据交互和同步。图3.2描述了当前三种主要的协同仿真主算法的活动图。在接下来的内容中，我们对这三种算法进行简单的介绍。

固定步长算法

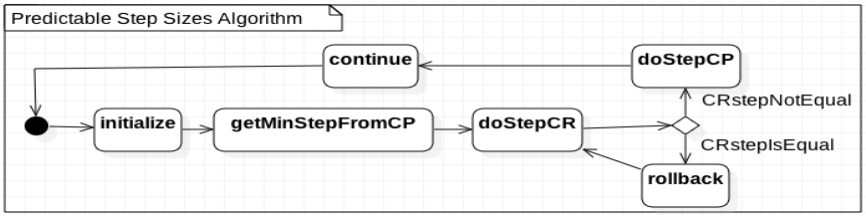
对于固定步长算法 [**?**]，所有的 FMU 都有一个相同的步长。当协同仿真主算法在 t 时刻调用 $doStep$ 函数执行步长 h 时，所有的 FMU 将从 t 时刻执行 h 步长并到达 $t+h$ 时刻。在执行下一个 $doStep$ 函数之前，要确保所有的 FMU 都执行完了上一个步长并且完成了数据交换。固定步长算法的活动图如图3.2(a)所示，该活动图主要包含三个活动：$initialize$, $doStep$ 和 $continue$，首先对所有的 FMU 进行初始化，之后调用 $deStep$ 函数进行仿真，最后在 $continue$ 活动中完成 FMU 的数据交换。在固定步长算法中，只要保证每个 FMU 的仿真是可靠的，则可以保证整个仿真过程的可靠性。但是，如果在某个 FMU 的仿真过程中出现错误，则会导致整个协同仿真过程出错。为了克服固定步长算法的缺陷，出现了可回滚的算法。



(a) 固定步长算法



(b) 可回滚算法



(c) 可预测步长算法

图 3.2: 三种协同仿真主算法的活动图

可回滚算法

FMI2.0 相对于 FMI1.0 添加了几个重要的特性，它支持了对于 FMU 状态的保存和回滚。例如，主算法调用 FMU_1 和 FMU_2 的 $doStep$ 函数执行 h 步长，但是 FMU_1 接受了该步长且 FMU_2 拒绝了该步长，则 FMU_1 和 FMU_2 都会执行 h 步长然后回滚到上一个时间点。可回滚算法的活动图如图3.2(b)所示，相比固定步长算法，可回滚算法要求所有的 FMU 支持 $rollback$ 方法, 且当某个 FMU 拒绝该步长时，所有的 FMU 将会回滚到上一个步长执行完之后的时间点。

可预测步长算法

如果可以预测 FMU 下一步能执行的最大步长（FMU 在不错过事件时，能执行的仿真的最长时间），则可以大大提高协同仿真的效率。因此，论文 [?] 提出了 $GetMaxStepSize$ 函数来预测 FMU 下一步能执行的最大步长，有了该函数的支持，就出现了可预测步长算法。图3.2(c)为可预测步长算法的活动图，首先，所有的 FMU

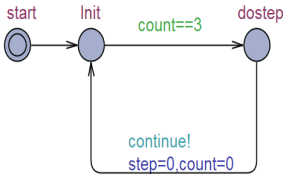
进行初始化，然后支持 $GetMaxStepSize$ 函数的 FMU 在 $getMinStepFromCP$ 活动中预测他们下一步能执行的最大步长，并且在所有的最大步长中选择最小的一个 h 作为所有 FMU 下一步执行的步长，之后所有的 FMU 执行 h 步长，如果所有的 FMU 都接受了该步长，则所有的 FMU 仿真该步长然后进行下一步。如果有 FMU 拒绝了该步长，则将所有的 FMU 回滚到上一个时间点，然后选择一个更小的步长进行仿真。

3.2.2　协同仿真主算法的建模和验证

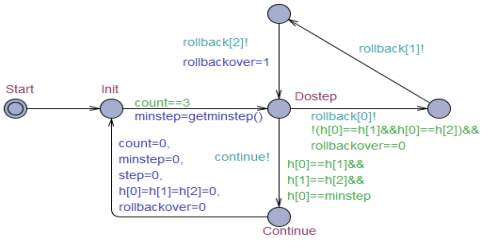
我们用时间自动机将三种不同类型的主算法进行形式化建模，图3.3是三种主算法的时间自动机模型。固定步长算法的时间自动机模型包含 $Init$ 和 $doStep$ 两个状态，且与 FMU 通过 $continue$ 信道进行通信。可回滚算法包括 $Init$、$DoStep$ 和 $Continue$ 三个状态。如果所有的 FMU 都接受了下一步要进行仿真的步长，则可回滚算法对应的时间自动机将发送 $continue$ 信号，且迁移到 $Continue$ 状态；否则，将发送 $rollback$ 信号，且返回到上一个状态。可预测步长算法包括 $Init$, $find\_CP\_MIN$, $DoStep$, $writeCP$ 四个状态。首先他们获得支持预测步长算法的多个 FMU 的最大步长，然后取所有最大步长的最小值 $step1$ 。然后执行该步长，如果所有的 FMU 都接受则发送 $continue$ 信号并迁移到下一个状态，否则发送 $rollback$ 信号并回滚到 $DoStep$ 状态。

UPPAAL 是基于时间自动机理论对实时系统进行验证的工具，其中使用到的时间自动机模型增加了整型变量、多种数据类型及同步信号等扩展。我们使用 UPPAAL 工具验证了这三个模型的可达性、活性及死锁。具体的验证属性如下所示：

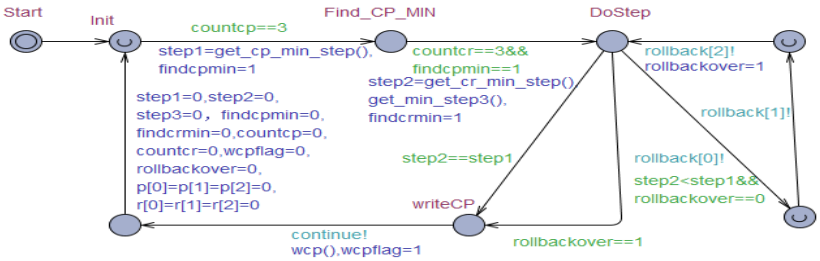
- $E\langle\rangle master.dostep$, $E\langle\rangle master.Continue$ and $E\langle\rangle master.writeCP$ 是可达性验证，用来验证这些系统状态是否可达；
- $master.Init->master.dostep$, $master.Init->master.Continue$ and $master.Init->master.Continue$ 是活性验证。如果主算法可以到达前一个状态，那么它最终也会到达后一个状态。



(a) 固定步长算法的时间自动机模型



(b) 可回滚算法的时间自动机模型



(c) 可预测步长算法的时间自动机模型

图 3.3: 三种不同主算法的时间自动机模型。

- $A[]notdeadlock$ 是死锁的验证, 用来验证主算法是否存在死锁。

实验的结果如表3.1所示，从表中我们可以看出不存在死锁，且可达性和活性都满足，说明我们的主算法是正确的。例如：属性 $A[]$ not $deadlock$ 满足，说明主算法不存在死锁；属性 $E\langle\rangle$ $master.doStep$ 满足，说明系统最终会到达 $doStep$ 状态。总结来说，我们在这一小节验证了我们所用到的协同仿真的主算法的正确性。

3.3　异构系统协同行为的验证

在本小节，我们使用一个水箱的案例对整个协同行为的验证过程进行详细的描述。由于，在验证过程中，需要用时间自动机对 FMU 进行形式化描述，我们首先给出了从 FMU 到时间自动机的映射规则，然后我们使用 SysML 对整个系统进行架构设计，之后基于 FMI 标准对系统的各个组件进行实现，并给出多个 FMU 之间的连接关系配置，最后我们用时间自动机建模了 FMU 的形式化模型，并输入到 UPPAAL 工具中针对验证属性进行验证分析。

表 3.1: 主算法的验证实验结果

主算法	验证属性	结果
固定步长	$A[]\ not\ deadlock$	True
	$master.Init- > master.dostep$	True
	$E\langle\rangle\ master.dostep$	True
可回滚	$A[]\ not\ deadlock$	True
	$master.Init- > master.Continue$	True
	$E\langle\rangle\ master.Continue$	True
可预测	$A[]\ not\ deadlock$	True
	$master.Init- > master.writeCP$	True
	$E\langle\rangle\ master.writeCP$	True

3.3.1 FMU 到 TA 的映射

在本文的第二章的预备知识中，我们给出了 FMU 和时间自动机的语法及语义，下面我们根据第二章的语法语义给出 FMU 和时间自动机的映射关系。我们发现 FMU 和时间自动机的语义之间存在一定的关联。FMU 的语义关注于 FMU 的执行序列，也就是状态随着时间的不断迁移；因此，时间自动机的执行迹跟 FMU 的执行序列十分相似，也是状态随着时间的迁移序列。因此，我们使用时间自动机来对 FMU 进行形式化描述，从而来分析多个 FMU 之间的协同行为。给定一个 FMU$F = (S, U, Y, D, s_0, set, get, doStep)$, 我们可以根据他们执行语义之间的关联，将 FMU 用时间自动机 $A = (L, l_0, E, I)$ 进行形式化描述：

- L 是时间自动机的有限位置集合。因此，时间自动机语义模型 L_A 中的状态可以看做为 FMU 中的状态，即 $(l, v) \Rightarrow s$。
- 时间自动机语义模型 L_A 中的初始状态可以看做为 FMU 中的初始状态, 即 $(l_0, v_0) \Rightarrow s_0$。
- FMU 中的输入变量 $u \in U$ 可以看做为时间自动机的 $Act_i \cup \{absent\}$。
- FMU 的输出变量 $y \in Y$ 可以看做为时间自动机的 $Act_o \cup \{absent\}$。

- 时间自动机的输入动作 $e \in Act_i$ 可以看做 FMU 中的 set 函数。
- 时间自动机的输出动作 $e \in Act_o$ 可以看做为 FMU 中的 get 函数。
- 时间自动机之间依靠 channel 的同步可以看做为 FMU 之间的依赖关系。$(u, y) \in D$ 表示输出变量 y 依赖于输入变量 u，在时间自动机中输出动作同样依赖于输入动作。
- 对于时间自动机，一个动作 $e \in Act$ 会触发一个迁移 $s \xrightarrow{e} s'$, 这个过程就相当于 FMU 里面的 $doStep$ 执行，会使其到达下一个状态。例如：时间自动机的迁移 $l \xrightarrow{e} l'$ 可以描述 FMU 的 $doStep(s, h)$ 被调用，并且此 FMU 接受了步长 h 并到达了下一个状态 s'；然而，此 FMU 也可能会拒绝步长 h, 并且发生了回滚，这个过程在时间自动机里面可以用一条边 $l' \xrightarrow{e} l$ 来进行描述，来表示回滚到了上一个状态。

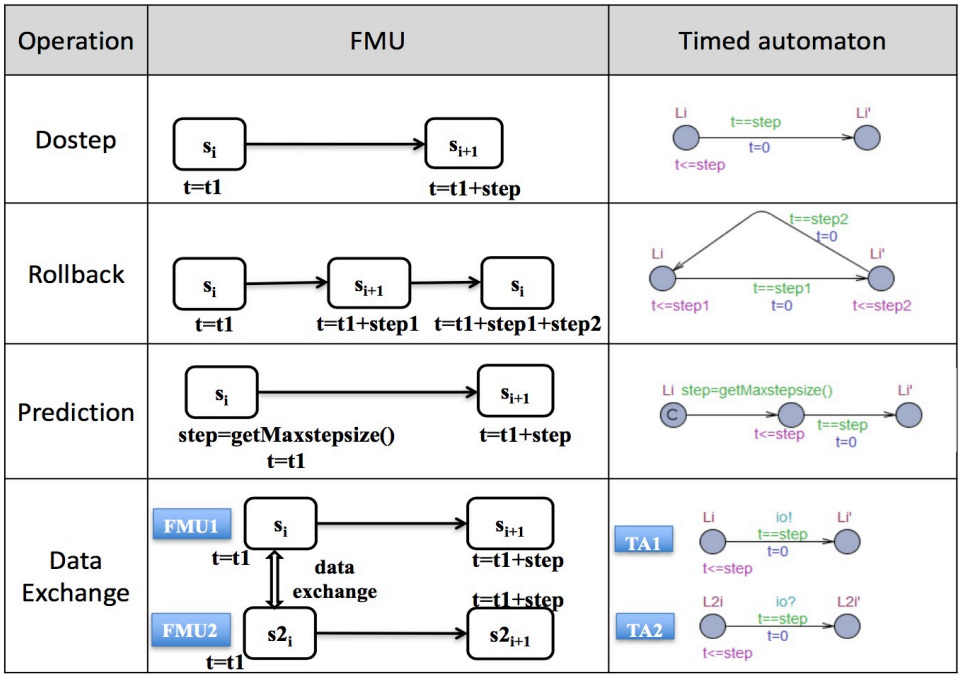


图 3.4: Encoding rules from FMU to TA.

将 FMU 直接转化为时间自动机是比较难得，S. Tripakis 层在论文 [10] 中将时间自动机编码为 FMU。我们受到此论文的启发，根据时间自动机和 FMU 之间语义

的关联，提出了几条从 FMU 到时间自动机在操作语义上的映射规则如图3.4所示。

给定 FMUt_1 时刻的当前状态 s_i，操作函数 $Dostep$ 可以使得 FMU 在 $t_1 + step$ 时刻到达 s_{i+1} 状态。这个操作可以用时间自动机的迁移来进行表示：在位置 L_i 延迟 $step$ 的时间并迁移到一个新的位置 L_i'。

对于操作函数 $Rollback$，给定 FMUt_1 时刻的当前状态 s_i，FMU 首先执行步长 $step1$ 并在 $t_1 + step1$ 时刻到达 s_{i+1} 状态，然后操作函数 $rollback$ 又使得 FMU 回滚到上一个状态 s_i。对于时间自动机来说，这个可以表示为在 L_i 位置延迟了 $step1$ 时间单位并迁移到新的位置 aL_i'，之后又迁移到了上一个位置 L_i。

对于操作函数 $Prediction$,给定一个状态 s_i,FMU 可以由一个函数 $getMaxstepsieze$ 得到下一步能执行的最大步长，然后执行此步长并在 $t_1 + step$ 时刻到达 s_{i+1} 状态。对于时间自动机，可以表示为在 L_i 位置执行了一个函数并得到要延迟的时间 $step$, 然后延迟该时间单位并迁移到一个新的位置 L_i' .

对于操作函数 $DataExchange$,两个 FMU 在 t_1 时刻的 s_i 状态执行 $DataExchange$ 进行了数据交换, 然后他们执行相同的步长并迁移到下一个位置 s_{i+1} 和 $s2_{i+1}$。对于时间自动机，可以表示为两个时间自动机在 t_1 时刻通过信道 io 进行了同步，然后延迟了相同的时间并 L_i 位置迁移到 L_{i+1} 位置。

为了将 FMU 用时间自动机进行形式化描述，我们提出了以上操作语义的映射规则。为了证明我们以上规则的正确性，我们分析了 FMU 和时间自动机的执行片段如下所示：

- 对于操作函数 $Dostep$ 的映射，在 FMU 和时间自动机中的执行片段为 (s_i, t_1), $(s_{i+1}, t_1 + step)$ 和 (l_i, t), $(l_i', t + step)$. 这说明时间自动机和 FMU 都执行 $step$ 的时间单位并到达了一个新的状态或是位置。

- 对于操作函数 $Rollback$ 的映射, 在 FMU 和时间自动机的执行片段为 (s_i, t_1), $(s_{i+1}, t_1+step1)$, $(s_i, t_1+step1+step2)$ 和 (l_i, t), $(l_i', t+step1)$, $(l_i, t+step1+step2)$。这说明时间自动机和 FMU 都首先执行了 $step1$ 时间单位，并且到达了一个新的状态或位置，然后执行了 $step2$ 时间单位, 又返回到了之前的状态或位置。

- 对于操作函数 $Prediction$ 的映射，FMU 和时间自动机的执行片段为 (s_i, t_1), $(s_{i+1}, t_1 + step)$ 和 (l_i, t), $(l_i', t + step)$. 这说明时间自动机和 FMU 都成功预测到了步长 $step$ 并执行了此步长。

- 对于操作函数 $DataExchange$ 的映射，FMU1 和时间自动机 TA1 的执行片段为 (s_i, t_1), $(s_{i+1}, t_1 + step)$ 和 (l_i, t), $(l_i', t + step)$。FMU2 和时间自动机 TA2 的执行片段为 $(s2_i, t_1)$, $(s2_{i+1}, t_1 + step)$ 和 $(l2_i, t)$, $(l2_i', t + step)$。这说明时间自动机和 FMU 在经过了数据交换后又执行了 $step$ 时间单位。

为了更加准确的验证映射规则的正确性，我们分析了时间自动机和 FMU 的整个执行序列。我们通过分析得到 FMU 和时间自动机的执行序列是等价的，验证了映射规则的正确性。在之后几个小节的内容中，我们将此映射规则应用到了水箱的案例中，根据我们之后对案例仿真片段的分析，我们也发现映射规则是正确的。

3.3.2　基于 SysML 的架构建模

为了更好的阐述我们的方法，我们采用了水箱的案例 [**?**] 作为驱动以更加形象的展示我们的方法。下面我们先简单的介绍一下水箱案例，然后用 SysML 建模语言对整个案例的架构建模。水箱案例：有一个水源可以向水箱里面注水，且水箱里面的水通过一个管道流入到一个水池当中，这个水源的流出由一个阀门控制，而阀门的开关由一个控制器进行控制。在本小节，由于水箱、阀门、控制器连接方式的不同，我们在此描述了三种不同类型的水箱系统。

我们用 SysML 来建模该系统的架构，其中用 SysML 的 BDD 图来描述系统中各个组件的结构，用 SysML 的 IBD 图用来描述各个组件之间的关联关系。各个组件的接口由连接器进行关联，因此系统各个组件的依赖关系可以用各个块的接口之间的连接进行表示。

图3.5是水箱案例的 SysML BDD 图，其中包含三个块：$Valve$, $Tank$ 和 $Controller$。$Valve$ 和 $Tank$ 是物理组件；$Controller$ 是信息组件。每一个组件都有自己的输入和输出接口，例如：$Valve$ 的输入接口是 vin，它用来输入阀门的开关 $OpenClosed$

信号。

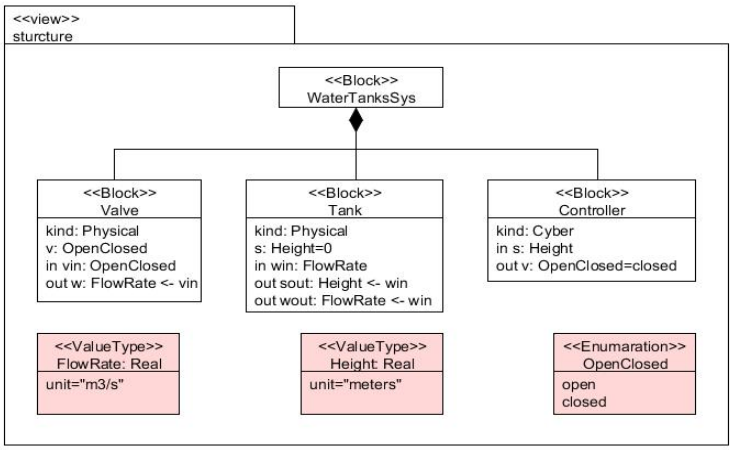


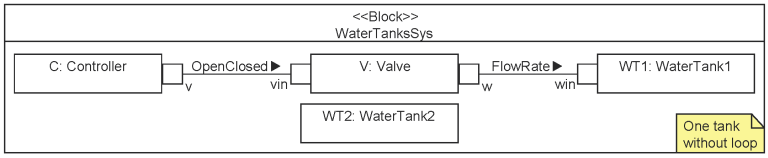
图 3.5: 水箱系统的 SysML BDD

图3.6 是 SysML 的 IBD 图，在这里我们给出了三种连接情形。在第一种情形中，系统包含一个阀门、一个控制器和一个水箱，控制器随机的给阀门发送开关信息，导致水箱中的水位不断的变化；在第二种情形中，控制器信号的发送受到水箱水位的影响，控制器根据水箱的水位发送开关信号；在第三种情形中，我们添加了一个水箱 $waterTank2$，水箱 $waterTank1$ 中的水通过管道先流入 $waterTank2$ 中，最后流入水池。

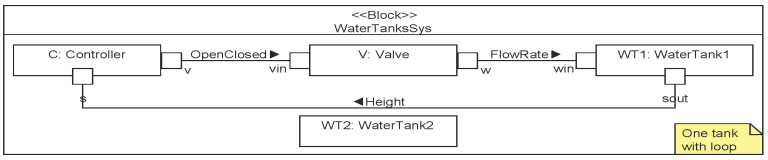
在本小节我们用 SysML BDD 图描述了系统组件的结构，并用 SysML 的 IBD 图描述了各个组件之间的连接。在下一个小节，我们基于 FMI 标准用 FMU 来实现每一个系统组件，并将 SysML IBD 中描述的关联关系用一个 FMU 之间的接口配置文件进行表示。

3.3.3　基于 FMI 的模型设计

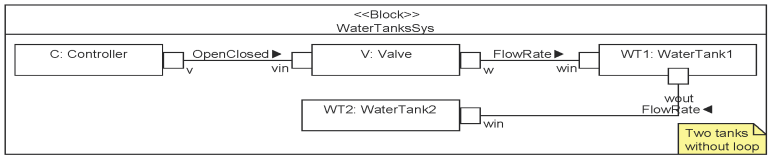
图3.7描述的是水箱系统中的 FMU 组件及各个 FMU 之间的连接。根据图3.6给定的 SysML IBD，我们也得到了三种 FMU 的情形。第一种情形如图3.7(a)所示，系统中有三个 FMU 组件：$Controller$, $Valve$ 和 $WaterTank1$ 和两个接口 v_vin 及 w_win。$Controller$ 和 $Valve$ 由 v_vin 接口连接；$Valve$ 和 $WaterTank1$ 由 w_win



(a) 情形 1



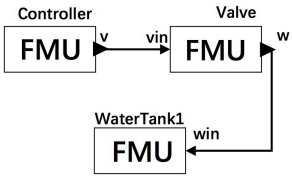
(b) 情形 2



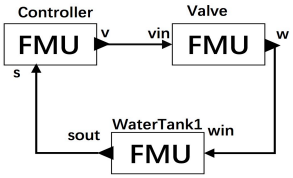
(c) 情形 3

图 3.6: 水箱系统的 SysML IBD

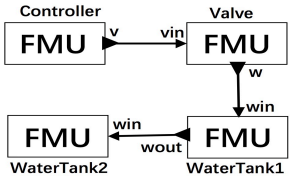
接口连接。第二种情形如图 3.7(b)所示，其中在第一种情形上添加了 $WaterTank1$ 和 $Controller$ 的接口 $sout\_s$，表示控制器信号的发送受到水箱 $WaterTank1$ 的水位影响。图3.7(c)是第三种情形，在第一种情形上添加了另一个水箱 $WaterTank2$, 水箱 $WaterTank1$ 和 $WaterTank2$ 由接口 w_out 连接。



(a) 情形 1



(b) 情形 2



(c) 情形 3

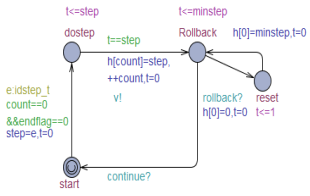
图 3.7: 水箱系统中 FMU 的三种连接情形

在本小节我们设计了 FMU 及 FMU 之间的连接，我们只需要添加一个协同仿真的主算法就可以在协同仿真引擎当中对整个异构系统进行协同仿真并得到仿真迹。但是，在进行仿真之前，我们要保证各个 FMU 之间协同行为的正确性，因此，我们在下一个小节基于时间自动机理论对系统的协同行为进行验证，这也是我们

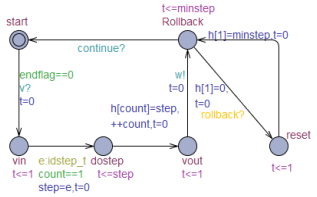
本文的主要贡献点之一。

3.3.4　协同行为的正确性验证

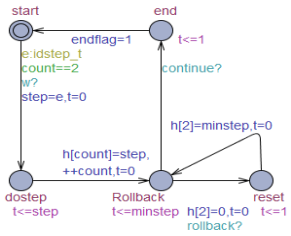
在本小节我们基于时间自动机理论对水箱系统中组件之间的协同行为的正确性进行验证。首先我们根据小节3.3.1中提出的映射规则，将水箱系统中的 FMU 用时间自动机进行形式化描述，并取小节3.2.2中建模的一个主算法（在此，采用可回滚算法作为主算法，其他算法的验证分析与该算法类似，在此不做描述），FMU 之间的接口配置文件我们用时间自动机之间的信道进行描述。由此，我们得到了一个由主算法、FMU 的时间自动机模型及由配置文件转化得到的信道组成的时间自动机网络。图3.8为上述小节中情形 1 的时间自动机网络模型。



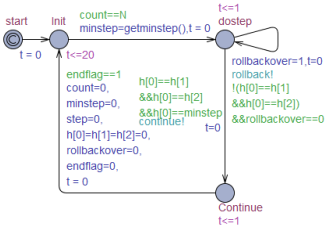
(a) FMU_controller 的时间自动机模型



(b) FMU_valve 的时间自动机模型



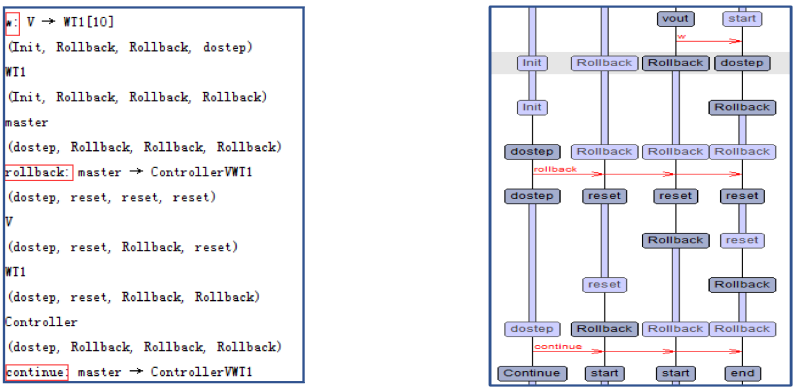
(c) FMU_WaterTank1 的时间自动机模型



(d) 可回滚主算法的时间自动机模型

图 3.8: 情形 1 的时间自动机网络: $controller \parallel valve \parallel WaterTank1 \parallel MA$.

图3.8(a), 3.8(b), 3.8(c)分别是 $controller$, $valve$ 和 $WaterTank1$ 的时间自动机模型。这些自动机都有四个主要的位置：$start$, $dostep$, $Rollback$ 和 $reset$。图3.8(a)是 $controller$ 的时间自动机模型，它首先通过信道 v 与 $valve$ 的时间自动机模型进行交互并到达 $Rollback$ 状态，然后等待主算法的信号，直到它收到了来自主算法



(a) 部分迹　　(b) 执行序列图

图 3.9: 协同过程在 UPPAAL 中的执行序列。

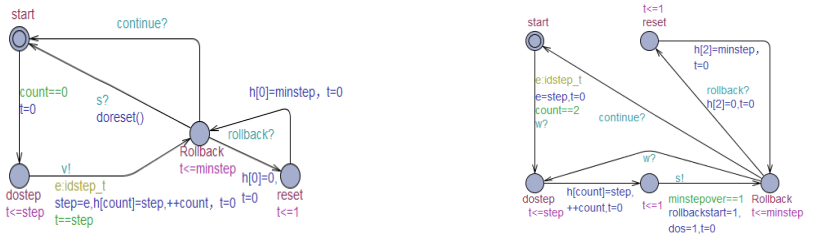
的 *continue* 信号再与其他的 FMU 进行数据交互并到达 *start* 状态；否则，它收到 *rollback* 信号，并回到 *Rollback* 状态。*valve* 和 $WaterTank1$ 的时间自动机中位置和迁移与 *controller* 类似。图3.8(d) 是主算法的时间自动机模型，首先主算法先进性参数初始化，然后根据条件来判断发出 *continue* 信号或是 *rollback* 信号。

图3.9是协同过程在 UPPAAL 中的执行片段，我们可以看到 *valve* 首先发送了信号 w 来与 $WaterTank1$ 进行数据交换，然后，$WaterTank1$ 到达了 *dostep* 状态，之后主算法广播 *rollback* 信号导致 FMU 到达 *reset* 状态，最后主算法发送 *continue* 信号使得所有的 FMU 到达 *start* 状态并开始下一步仿真。通过对执行序列的分析，我们发现模型可以正确仿真。

为了比较小节3.3.3中描述的三种情形的协同行为，我们同时用时间自动机形式化描述了其他两种情形。对于第二种情形，我们在第一种情形基础上添加了 *controller* 和 $WaterTank1$ 之间的信道 s 如图3.10所示；对于第三种情形我们建模了 $WaterTank2$ 的模型并添加了信道 $w2$ 如图 Fig.3.11所示。其他的模型与第一种情形中的模型类似，我们在此只给出了新增加的或有改动的模型。接下来我们将对这三种情形进行验证。

我们对每一种情形都验证了以下属性：

- $E\langle\rangle\, WT1.Rollback$ 和 $E\langle\rangle\, master.Continue$ 为可达性验证，它表示 $WaterTank1$



(a) TA for FMU_controller　　(b) TA for FMU_WaterTank1

图 3.10: TA for connection case 2.

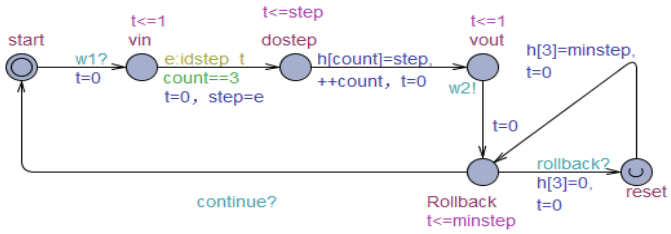


图 3.11: TA for FMU_WaterTank2 of connection case 3.

表 3.2: 三种情形的验证结果

情形	验证属性	结果
情形 1	$E\langle\rangle\ WT1.Rollback$	True
	$E\langle\rangle\ master.Continue$	True
	$master.start- > master.Continue$	True
	$A[]\ not\ deadlock$	True
情形 2	$E\langle\rangle\ WT1.Rollback$	True
	$E\langle\rangle\ master.Continue$	False
	$master.start- > master.Continue$	False
	$A[]\ not\ deadlock$	True
情形 3	$E\langle\rangle\ WT1.Rollback$	True
	$E\langle\rangle\ master.Continue$	True
	$master.start- > master.Continue$	True
	$A[]\ not\ deadlock$	True

将到达 $Rollback$ 状态且主算法会到达 $Continue$ 状态。

- $master.start- > master.Continue$ 为活性验证，它表示一旦主算法开始，它最早会到达 $Continue$ 状态.

- $A[]\ not\ deadlock$ 为死锁的验证，它用来验证系统有无死锁。

验证结果如表3.2所示，我们发现情形 1 和情形 3 的验证属性都满足，它表示该情形的协同是正确的。然而情形 2 的可达性和活性不满足，是由于在该模型中出现了环路依赖，我们需要消除该环路依赖再进行下一步的仿真，在本文中我们只关注协同行为的验证，对于如何消除环路依赖在接下来的工作中我们会做进一步研究。

3.4　本章小结

在本小节我们提出了一种新的方法来验证异构系统协同行为的正确性，我们首先用 SysML 建模语言建模了整个系统的架构，然后基于 FMI 标准对整个架构进行实现。之后提出了一种映射规则将 FMU 用时间自动机进行了形式化描述，最后基于时间自动机理论对整个系统的协同行为的正确性进行了验证。经过本小节的验证，我们可以得到通过验证的基于 FMI 标准的模型，该模型可以在协同仿真引擎中直接仿真并得到仿真迹，在下一个小节，我们将提出一种高效的统计模型检测方法，针对某些验证属性对协同仿真的迹进行定量的评估。

第四章　基于抽象和学习的分布式统计模型检测方法

通过对上一个章节得到的模型进行协同仿真，我们可以得到模型的仿真迹。将仿真迹和需要验证的属性输入到统计模型检测器中，即可以对整个异构系统的行为进行定量的评估分析。但是，由于基于统计模型检测的大型异构系统验证将产生大量的仿真迹，并且每一条迹的产生都十分的耗费时间。因此，使用统计模型检测算法算法对大型异构系统进行验证效率将十分低下。为了解决这一问题，在这一章我们提出了一种基于抽象和学习的分布式统计模型检测方法来提高统计模型检测的效率。本章的第一小节提出了本方法的技术路线及分布式架构设计；第二小节应用分布式架构对一种经典的统计模型检测算法-贝叶斯区间估计算法 [5] 进行了分布式实现；第三小节首先提出了基于抽象和学习的统计模型检测算法并给出了该算法的分布式实现和优化；最后一个小节给出了基于抽象和学习的分布式统计模型检测算法的算法分析。此外，该算法的效率及准确性对比我们在后面的实验章节给出。

4.1　技术路线及分布式架构设计

在本小节，我们首先给出本方法的技术路线及使用到的分布式架构设计。本方法从在技术上针对两个方面对统计模型检测算法进行改进，并使用了主从式的分布式架构。

4.1.1　技术路线

对统计模型检测算法的效率有直接影响的因素主要有两个：（1）验证过程中产生的仿真迹的数量；（2）产生单条仿真迹需要的时间。在本章，我们针对这两个

因素对统计模型检测的效率进行改进。图4.1是本章的技术路线图，统计模型检测器主要包含三个组件：*仿真器 (simulator)*、*统计模型检测算法 (SMC algorithm)* 和*模型验证器 (model checker)*。*仿真器*产生仿真迹并输入到*模型验证器*之中，*模型验证器*跟根据验证属性来验证该仿真迹是否满足该验证属性，并返回验证结果（返回 1 表示满足，0 表示不满足）。*统计模型检测算法*收集来自*模型验证器*的验证结果并进行统计分析，最终返回验证结果。为了减少验证过程中产生的仿真迹的数量，我们提出了使用抽象和学习的方法对统计模型检测算法进行改进（AL-SMC），经过多次实验分析，我们发现该方法可以有效减少验证过程中产生的仿真迹。根据论文 [26], 我们发现统计模型检测算法是可以基于主从架构进行分布式改进的，即可以用多个仿真器产生仿真迹并用一个统计模型检测器进行统计分析，因此，我们可以使用分布式技术来减少产生单条仿真迹需要的时间，并且我们基于分布式技术设计了分布式的贝叶斯区间估计算法。在本章中，我们结合分布式技术和 AL-SMC 技术提出了基于抽象和学习的分布式统计模型检测算法（DAL-SMC），该算法同时有效减少了验证过程中产生的仿真迹的数量和产生单条仿真迹需要的时间。

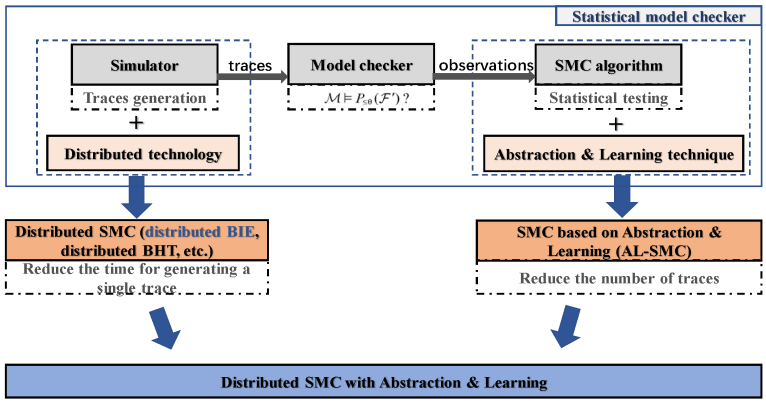
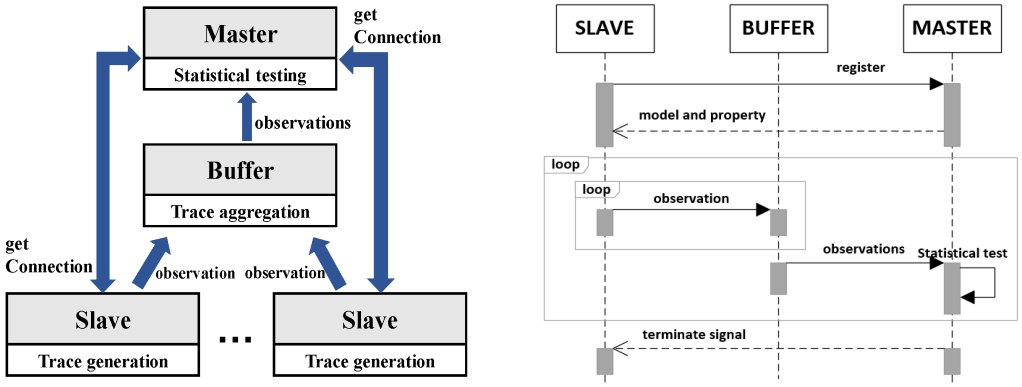


图 4.1: 基于抽象和学习的分布式统计模型检测方法技术路线

4.1.2　分布式架构设计

在本小节，我们给出了统计模型检测算法的分布式架构设计如图4.2(a)所示：多个从属机器进行模型的仿真并产生仿真迹，然后对该仿真迹进行验证并将验证结

果（observation）存储到缓冲器（Buffer）中，主机器从缓冲器中收集验证结果并进行统计分析。在对统计模型检测算法进行分布式改进时，有一个关键点就是要避免引入偏差。为了解决这个问题，我们使用了论文 [27] 提出的使用缓冲器和批量处理器来避免偏差的方法。



(a) 分布式架构　　(b) 通信协议

图 4.2: 分布式统计模型检测的架构设计

图4.2(b)是分布式统计模型检测的通信协议时序图：(i) 多个从属机器跟主机器建立连接，主机器将模型和验证属性发送给从属机器。(ii) 从属机器将验证结果发送到缓冲器中，缓冲器中存储的验证结果最终发送到主机器。(iii) 主机器进行统计分析，直到统计分析过程结束，主机器发送终止信号给从属机器，从属机器结束仿真。该分布式统计模型检测算法的框架具有通用性。基于本框架，我们在下个小节设计了分布式的贝叶斯区间估计算法，实际上我们可以将该框架应用到任意一个经典的统计模型检测算法为不只局限于贝叶斯区间估计算法。

4.2　分布式的贝叶斯区间估计算法设计

在本小节，我们首先介绍了统计模型检测中一个经典的算法-贝叶斯区间估计算法。之后我们将上一小节提出的分布式框架应用到本算法上，并设计了分布式的贝叶斯区间估计算法，使得贝叶斯区间估计算法产生单条仿真迹的时间有效减少，从而提高了贝叶斯区间估计算法的效率。

4.2.1　贝叶斯区间估计算法介绍

贝叶斯区间估计算法是可以用来高效的评估随机模型 M 满足验证属性 ϕ(即, $p = Prob(M \models \phi)$) 的概率区间。贝叶斯区间估计算法 (算法 1) 调用仿真器不断的产生仿真迹，然后验证它是否满足验证属性 ϕ 并返回验证结果，最后调用统计检测算法 (算法 2) 来计算估计概率 p', 概率区间范围 (t_0, t_1) 和后置概率 γ。直到满足条件 $\gamma >= c$, 算法终止并返回 t_0, t_1 和 p', 否则继续循环产生仿真迹并计算。

Algorithm 1 贝叶斯区间估计算法

Input: 验证属性 ϕ, 半区间大小 $\delta \in (0, 1/2)$, 区间覆盖系数 $c \in (1/2, 1)$, 前置 beta 分布的参数 α、β

Output: 概率区间 (t_0, t_1)、估计概率 p'

```
x = 0; n = 0
loop
    σ = draw a simulation of the model
    if σ ⊨ φ then
        x = x + 1
    end if
    n = n + 1
    t0,t1,p',γ = CallAlgorithm2(δ, α,β,x, n);
    if γ >= c then
        return t0,t1,p',γ
    end if
end loop
```

4.2.2　分布式贝叶斯区间估计算法

基于贝叶斯区间算法及我们提出的分布式框架，我们设计了分布式的贝叶斯区间估计算法。算法3是分布式贝叶斯区间估计算法的从算法。B 是批处理分区的大小，用来收集从属机器的仿真结果并存储起来，因此一个从属机器对应的一个批处理分区里面有多个仿真结果。该算法不断的调用仿真器产生仿真迹并验证该迹是否满足验证属性 ϕ。直到满足条件 $runs == B$, 该算法终止并返回满足属性的

Algorithm 2 统计检测算法

Input: 半区间大小 $\delta \in (0, 1/2)$, 前置 beta 分布的参数 α、β、满足验证属性迹的数量 x、产生迹的总数量 n

Output: 概率区间 (t_0, t_1)、后置概率 γ、估计概率 p'

1: p' = (x+α)/(n+α+β)
2: (t_0, t_1) = (p-δ,p+δ)
3: **if** $t_1 > 1$ **then**
4: (t_0, t_1) = (1-2δ,1)
5: **else if** $t_0 > 0$ **then**
6: (t_0, t_1) = (0,2δ)
7: **end if**
8: $\gamma = \int_{t_0}^{t_1} f(u|x_1, ..., x_n)du$

样本数量 $sats$，否则继续循环调用仿真并验证。算法4是分布式贝叶斯区间估计算法的主算法。K 是缓冲器的大小，用来缓冲来自某个仿真器的批处理分区，如果有某个仿真器产生迹的速度较快，可以将其批处理分区缓冲到该仿真器对应的缓冲区内，且存储的数量不能超过K。主算法的主要步骤如下：(i) 从属机器与主机器建立连接，主机器将模型和验证属性发送给从属机器。(ii) 主算法随机选择一个从算法，如果该从算法的从属机器对应的缓冲区的大小没有超过K，则调用该从算法。(iii) 如果缓冲器中所有的缓冲区都不为空，则获取缓冲区的验证结果并更新n 和 x 的值。(iv) 调用算法2来计算估计概率 p', 概率区间 (t_0, t_1) 和后置概率 γ. 直到满足条件 $\gamma >= c$, 算法终止并返回 t_0, t_1 和 p' 的值, 否则继续循环步骤 (ii)、(iii) 和 (iv).

4.3　基于抽象和学习的分布式统计模型检测算法

在上一小节，我们通过对贝叶斯区间估计算法进行分布式设计，减少了产生单条仿真迹需要的时间，一定程度上提高了统计模型检测的效率. 我们在小节4.1.1中提到，还有一个影响统计模型检测效率的关键因素是验证过程中产生的仿真迹的数量。在本小节，我们先使用抽象和学习的方法减少贝叶斯区间估计算法验证过程中产生的仿真迹的数量，然后将分布式框架应用到基于抽象和学习的统计模型

Algorithm 3 分布式贝叶斯区间估计算法的从算法

Input: 验证属性 ϕ, 模型 M, 批处理区大小 B

Output: 满足验证属性迹的数量 $sats$

```
1:  sats = 0, runs = 0
2:  loop
3:      σ := getSimulationTrace(M)
4:      if σ ⊨ φ then
5:          sats ++
6:      end if
7:      runs ++
8:      if runs == B then
9:          return sats
10:     end if
11: end loop
```

检测算法之中，从而从产生仿真迹的数量及产生单条仿真迹需要的时间两个方面提高了统计模型检测的效率。

4.3.1　基于抽象和学习的统计模型检测算法

贝叶斯区间估计算法有一个缺点：当它验证的属性的最终结果接近 0.5 时，将需要消耗大量的仿真迹。为了解决这个问题，我们使用抽象和学习的方法对贝叶斯区间估计算法进行了改进。在此，我们只对基于抽象和学习的统计模型检测算法给出简单的介绍，详细的信息见论文 [25]。图4.3本方法的框架图，该算法主要包含三个步骤: (i) 我们将模型仿真生成的迹输入得到抽象过程中，获得经过抽象的简化迹。该抽象过程又包含三个步骤：**基于属性的投影, 基于主成分分析算法的降维 [34] 和关键状态提取**. (ii) **前缀频率树的构建和约减**过程用到了论文 [35] 提出的学习方法将简化的迹构建成了一棵前缀频率树。经过该过程，我们实际上是将原始的模型划分为了多个**子模型**. (iii) **基于多个贝叶斯区间估计的评估分析**过程在多个子模型上分别执行贝叶斯区间估计算法来得到各个子模型的概率，然后将各个模型的概率累加即可得到最终概率。

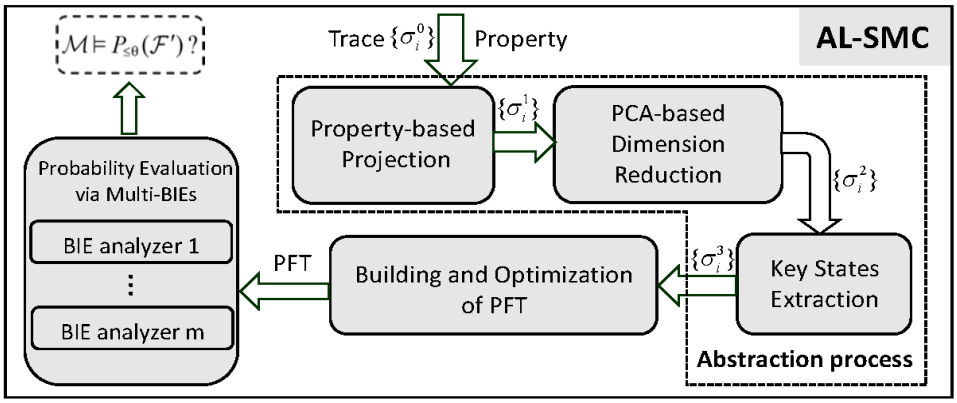


图 4.3: 基于抽象和学习的统计模型检测算法框架图

4.3.2　基于抽象和学习的分布式统计模型检测算法

相对于贝叶斯区间估计算法，基于抽象和学习的统计模型检测算法减少了验证过程中产生的仿真迹，在验证概率接近 0.5 时效果尤为显著。此外，我们发现该方法也可以应用本文提出的分布式框架来减少产生单条迹消耗的时间，因此我们将该方法进行了分布式设计，即得到了基于抽象和学习的分布式统计模型检测算法。该算法的主算法和从算法如算法6和算法7所示。算法6是基于抽象和学习的分布式统计模型检测算法的主算法，其中 SN 是抽象过程中用到的样本迹的数量。主算法包含三个主要的步骤：(i) 主算法调用仿真器产生多条样本迹 $\sigma[0...SN-1]$，并将其输入到抽象过程中来得到经过抽象简化的迹 $\sigma'[0...SN-1]$。然后，用简化的迹 $\sigma'[0...SN-1]$ 构建出前缀频率树并进行约减。(ii) 从属机器与主机器建立连接，主机器将模型，验证属性及前缀频率树发送给从属机器。(iii) 主算法随机选择一个从算法，如果该从属机器对应的缓冲区的值小于 K，则调用该从算法产生验证结果（相对于分布式贝叶斯区间估计算法，此时返回的是一个数组，存储着各个子模型对应的满足验证属性的批处理区的数量，而不再只是一个数值）并进行赋值。(iv) 如果所有的缓冲器都不为空，则调用算法2来计算各个子模型的估计概率 $p'[j]$ 和后置概率 $\gamma[j]$，如果某个子模型的后置概率满足 $\gamma[j] > c$，则对于该子模型的验证结束。(v) 直到所有子模型的验证都结束，则得到最终概率 $p' := \sum_{i=0}^{S-1} p[i]$, 否则继续循环步骤 (iii)、(iv) 和 (v).　算法7是基于抽象和学习的分布式统计模型检测算法

的从算法。T 是主算法中构建并约减之后的前缀频率树。d 是前缀频率树叶子节点的数量，也就是划分得到的子模型的数量；$sats[i]$ 是第 i 个子模型满足验证属性的迹的数量。从算法不断的调用仿真器产生迹 σ，并将其输入到抽象过程中产生抽象迹 σ'。如果该迹满足验证属性，则根据前缀频率树将该迹划分给某个子模型，并更新 $sats$ 数组。直到满足条件 $runs == B$，算法终止并返回数组 $sats[0...S-1]$，否则继续调用仿真器并循环。

相对于小节4.2中提出的分布式的贝叶斯区间估计算法，本算法得益于使用抽象和学习技术减少仿真迹的数量而更加高效。然而，该算法由于使用多个贝叶斯区间统计分析器而导致统计分析过程的误差会增大，在下面的小节，我们会首先参数优化的方法来减少误差，并给出该算法的复杂度及误差分析。

4.3.3　参数优化

在本小节我们用两种参数优化的方法来减少统计分析的误差：（1）我们发现模型划分出的子模型越均匀，统计分析的误差越小，因此我们用 r **优化**来将模型划分的更加均匀。（2）用 δ **预测**来预测各个子模型应该用到的贝叶斯区间估计算法的半区间大小。

r **优化**：在基于抽象和学习的统计模型检测当中，我们通过前缀频率树的构建和约减将模型划分为了多个子模型，其中前缀频率树的叶子节点的数量即为划分得到的子模型的数量。r 为前缀频率树的约减参数，即不同的 r 值会得到不同约减程度的前缀频率树，模型也就可能会被划分为不同数量的子模型。然而，r 值的确定并不是十分容易。一方面，如果 r 的值太大，我们将会得到大量的子模型导致统计分析的误差变大; 另一方面，如果 r 的值太小，我们将会得到少量的子模型导致算法的效率变低。为了更好的确定 r 的值，我们采用了遗传算法 [**?**] 来获取最优的 r 值。

$$r_k = M - \sum_{i=1}^{k}\sum_{j=1}^{k}|T_i - T_j| \tag{4.1}$$

假设 k 为遗传算法的种群数量，也就是在每一代划分的子模型的数量；T_i 是

第 i 个子模型的满足验证属性的迹的数量。为了得到最优的 r 值，我们需要将模型划分的尽量均匀并要保证划分出的子模型的数量适中。因此，我们采用公式4.1 和公式 $k \geq 6 \&\& k \leq 10$ 分别作为遗传算法的作为评估函数和约束函数。M 是一个较大的数值，我们划分越均匀，则 $\sum_{i=1}^{k}\sum_{j=1}^{k}|T_i - T_j|$ 的值越小。因此，我们可以根据以上的方法得到最优的 r 值。

δ **预测**: 经过前缀频率树的构建和约减，我们得到了多个子模型，接下来我们就可以在多个子模型上使用贝叶斯区间估计算法，每个贝叶斯区间估计算法都有自己的半区间大小 δ 和区间覆盖系数 c。在算法7中, 我们假设所有的贝叶斯区间估计算法都有相同的 δ 和 c 值。然而，各个子模型最终得到的估计概率可能会有比较大的差异。例如：我们假设所有子模型上应用的贝叶斯区间估计算法的 δ 值都是 0.1, 但是可能有的子模型最终验证得到的估计概率本来就比较小，可能还不到 0.1，此时对于该子模型来说，它应用的贝叶斯区间估计算法的 δ 值显然就不合理，并且最终会导致一个比较大的统计误差。为了解决这一问题，我们用公式4.2所示来预测 δ 的值。其中，k 是子模型的数量；T_m 是第 m 个子模型满足验证属性的迹的数量；N_i 是第 m 个子模型上产生的总的仿真迹的数量；η 是半区间比率, 本质上只是一个可以定义的参数，η 越大，得到的最终估计概率约准确。应用公式4.2来预测 δ 之后，使各个子模型应用的贝叶斯区间估计算法得到的估计概率更加准确。

$$\delta_m = T_m / \sum_{i=1}^{k} N_i / \eta \tag{4.2}$$

4.4　算法分析

4.4.1　时空复杂度分析

基于抽象和学习的分布式统计模型检测算法的时空复杂度如表4.1所示。

(i) 抽象过程的时空复杂度分别为 $O(mn) + O(min(k^3, n^3))$ 和 $O(k^2)$，其中 m 表示迹的长度；n 表示样本迹的数量；k 表示迹的维数。

(ii) 前缀频率树构建和约减过程的时空复杂度分别为 $O(mn)$ + $O(d \log d))$ 和 $O(\log d)$，其中 d 表示前缀频率树叶子节点的数量。

表 4.1: 时空复杂度

算法过程	时间	空间
迹的抽象	$O(mn)$+$O(min(k^3, n^3))$	$O(k^2)$
前缀频率树	$O(mn)$+$O(d \log d))$	$O(\log d)$
统计分析	$O(B * (\log d + i))$	$O(1)$

(iii) 假设贝叶斯区间估计算法迭代次数为 B，一次统计分析的时间复杂度为 $O(i)$ 且划分子模型需要的时间复杂度为 $\log d$。则统计分析的时间复杂度为 $O(B * (\log d + i))$。

4.4.2 误差分析

对于一特定的验证属性，模型在算法6的步骤（i）和（ii）阶段在验证概率上是等价的。因此，误差只出现在统计分析阶段，我们使用了多个贝叶斯区间估计来进行统计分析，假设每个贝叶斯区间估计算法的半区间大小是 δ 且子模型的数量为 M，则此时算法的误差将不会超过 $\delta * M$。此外，在小节4.3.3中，我们还使用 δ 预测的方法进行了算法优化，因此最终经过优化的算法的误差 (ξ) 满足公式4.3。

$$|\xi| \leq \sum_{m=1}^{M} (T_m / \sum_{i=1}^{k} N_i / \eta) \tag{4.3}$$

4.5 本章小结

本章节将上一章节经过协同仿真得到的仿真迹作为输入，使用统计模型检测算法对仿真迹进行评估分析，但是由于验证过程需要消耗大量的仿真迹且协同仿真产生一条迹本身就十分耗时。因此，我们提出了一种基于抽象和学习的分布式统计模型检测算法来提高统计模型检测的效率，并采用参数优化的方法对该算法的误差问题进行了解决。我们已经将协同仿真引擎及改进的模型检测算法在工具中进行了实现，在下一个章节我们会给出我们工具的介绍，而对于本改进算法的效率及误差评估将会在本文最后的实验部分给出对比分析。

Algorithm 4 分布式贝叶斯区间估计算法的主算法

Input: 验证属性 ϕ, 模型 M, 半区间大小 $\delta \in (0, 1/2)$, 区间覆盖系数 $c \in (1/2, 1)$, 前置 beta 分布的参数 α、β, 缓冲区大小 K, 批处理区大小 B, 从属节点数量 N

Output: 概率区间 (t_0, t_1)、估计概率 p'

```
1: x = 0; n = 0
2: batch[0...N-1][0...K-1], buffer[0...K-1], slave[0...N-1]
3: loop
4:    getConnection(slave[i])
5:    sendModelandProperty(M,φ,slave[i])
6:    if i>N-2 then
7:       return
8:    end if
9: end loop
10: loop
11:    node = random(0,N-1)
12:    if buffer[node] < K then
13:       batch[node][buffer[node]] = CallAlgorithm3(node)
14:       buffer[node]++
15:    end if
16:    if forall(i < N)buffer[i] > 0 then
17:       loop
18:          x += batch[i][0]
19:          n += B
20:          buffer[i]−
21:          batch[i][buffer[i]] = 0
22:          if i > N − 1 then
23:             return
24:          end if
25:       end loop
26:    end if
27:    t0,t1,p′,γ = CallAlgorithm2(δ, α,β x, n)
28:    if γ >= c then
29:       return  t0,t1,p′
30:    end if
31: end loop
```

Algorithm 5 基于抽象和学习的分布式统计模型检测算法的主算法预处理

Input: 验证属性 ϕ, 模型 M, 从属机器节点数量 N，抽象过程中产生的样本迹的数量 SN

1: $\sigma[0...SN-1]$ = getSampleTraces(M)
2: $\sigma'[0...SN-1]$ = abstractTraces($\sigma[0...SN-1]$)
3: T = ConstructPFT($\sigma'[0...SN-1]$)
4: d = getSubSpacesNum(T)
5: **loop**
6: getConnection(slave[i])
7: sendModelandPropertyandPFT(M,ϕ,T,slave[i])
8: **if** i>N-2 **then**
9: **return**
10: **end if**
11: **end loop**
12: x[0...d-1], n
13: batch[0...N-1][0...K-1][0...d-1], buffer[0...K-1], slave[0...N-1]

Algorithm 6 基于抽象和学习的分布式统计模型检测算法的主算法

Input: 验证属性 ϕ, 模型 M, 半区间大小 $\delta \in (0, 1/2)$, 区间覆盖系数 $c \in (1/2, 1)$, 前置 beta 分布的参数 α、β, 缓冲区大小 K, 批处理区大小 B, 从属机器节点数量 N，抽象过程中产生的样本迹的数量 SN

Output: 前缀频率树 T, 估计概率 p'

CallAlgorithm5
loop
  node = random(0,N-1)
  if $buffer[node] < K$ **then**
    batch[node][buffer[node]][0...d-1] = **CallAlgorithm7(node)**
    buffer[node]++
  end if
  if $forall(i < N) buffer[i] > 0$ **then**
    loop
      x[0...d-1] += batch[i][0][0...d-1]
      n += B
      buffer[i]−
      batch[i][buffer[i]] = Φ
      if $i > N - 1$ **then**
        return
      end if
    end loop
  end if
  ter[0...d-1] = false
  j = findUnTerminateBIE(ter[0...d-1])
  $p'[j]$,$\gamma[j]$ = **CallAlgorithm2(**δ**,** α,β $x[j]$**,** n**)**
  if $\gamma[j] > c$ **then**
    $ter[j] = true$
  end if
  if forall(ter[i] ∈ ter[0...d-1])==true **then**
    return
  end if
end loop
return $p' = \sum_{i=0}^{d-1} p'[i]$

Algorithm 7 基于抽象和学习的分布式统计模型检测算法的从算法

Input: 验证属性 ϕ, 模型 M, 批处理区大小 B，前缀频率树 T, 子模型数量 d

Output: 各个子模型对应的满足验证属性的迹的数量 $sats[0...d-1]$

1: $sats[0...d-1]$, $runs$ = 0
2: **loop**
3: σ = getSimulationTrace(M)
4: σ' = abstractTrace(σ)
5: **if** $\sigma \models \phi$ **then**
6: i = findSubSpace(σ', T)
7: $sats[i]$ ++
8: **end if**
9: $runs$ ++
10: **if** $runs$ == B **then**
11: **return** $sats$
12: **end if**
13: **end loop**

第五章　工具实现

5.1　异构系统验证分析工具（Co-SMC）介绍

在本文我们提出了一种面向异构系统的验证分析方法，该方法基于协同仿真和统计模型检测，实现对异构系统行为的验证、分析。为了对该方法提供支持，我们设计、开发了异构系统验证分析工具，该工具主要由协同仿真、统计模型检测器及统计分析器三部分组成。在协同仿真器模块中，用户可以输入要进行协同仿真的多个 FMU 模型，然后设置仿真步长和多个 FMU 模型之间的参数对应关系，最后，可以从 4 种协同仿真算法中选择适当的协同仿真算法，对系统模型进行协同仿真并产生协同仿真的迹。统计模型检测器模块以协同仿真器产生的迹为输入，然后设置需要验证的属性，最后选择适合的统计模型检测算法进行定量的验证分析。用户可以在统计分析器模块对协同仿真器产生的迹或是统计模型检测器产生的验证结果进行统计分析，并绘制出饼状图、折线图等可视化的图形，有利于对模型的各种行为进行更加直观的分析。接下来我们对 Co-SMC 工具的各个部分进行详细的描述。 图5.1是 Co-SMC 工具的协同仿真器，该工具主要包括

5.2　Co-SMC 程序实现

5.3　本章小结

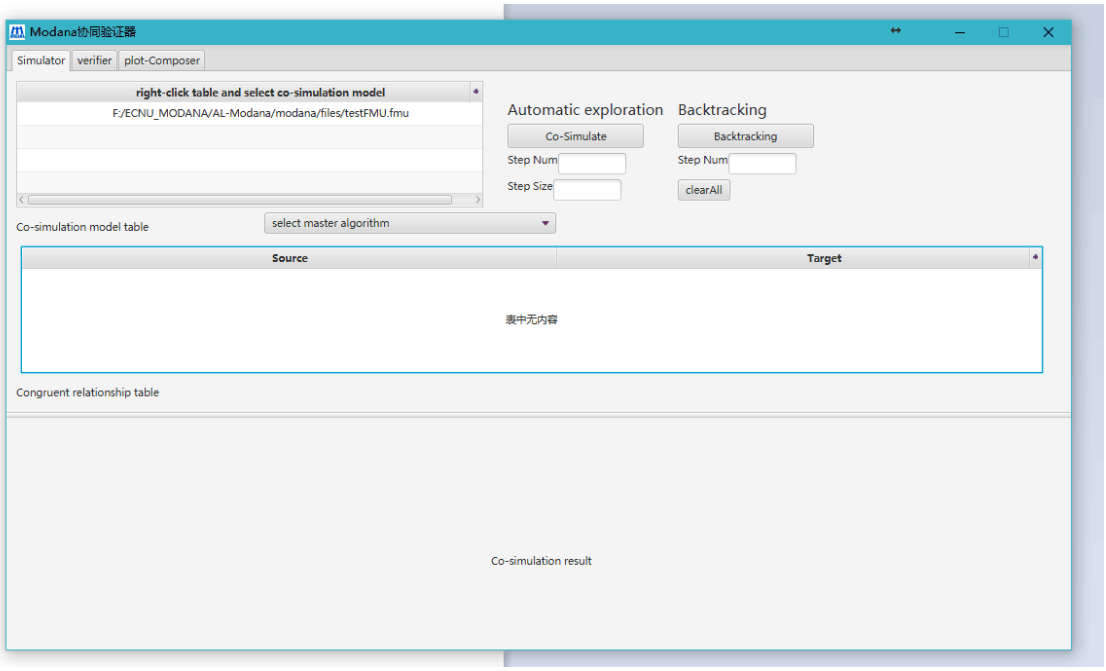


图 5.1: Co-SMC 协同仿真器

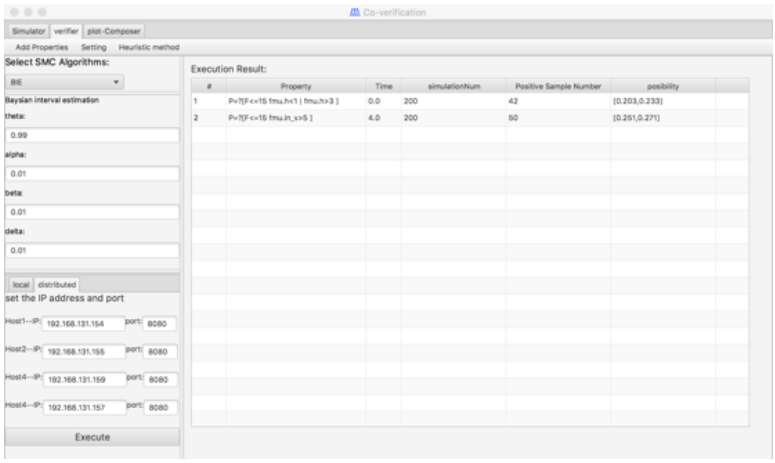


图 5.2: Co-SMC 统计模型检测器

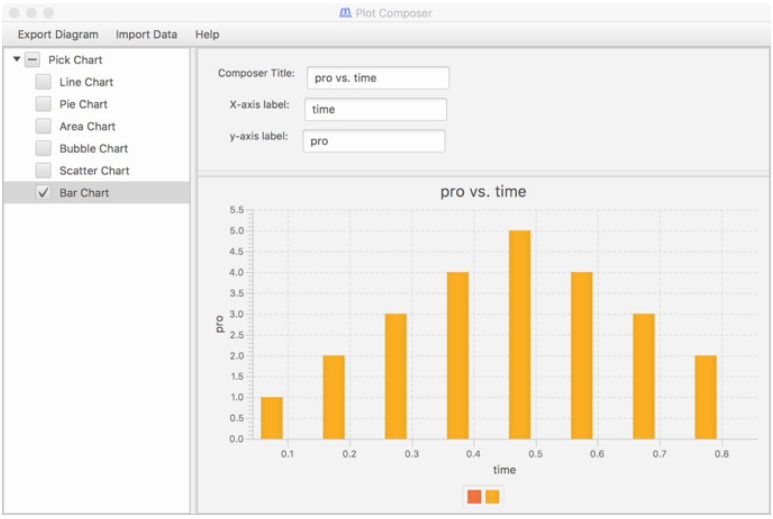


图 5.3: Co-SMC 协同仿真器

第六章　案例分析与实验评估

在本小节，我们使用两个案例（智能温控系统和机器人路径规划系统）来展示本文提出的方法的有效性。首先，我们使用 SysML 建模语言对这案例进行建模并根据基于 FMI 标准进行实现以得到基于 FMI 标准的仿真模型，然后我们使用章节 3 提出的方法对系统行为的正确性进行验证，之后将验证通过的模型进行协同仿真并将得到仿真迹用章节 4 提出的方法进行评估分析得到最终的评估结果。由于在章节 3 中我们已经使用了水箱的对整个方法进行了详细的描述，因此在本章节的案例中我们不对章节 3 涉及的过程做过多的描述，而是重点关注模型通过统计模型检测算法验证之后得到的评估结果及章节 4 提出的经过改进的算法的准确度及效率问题。

6.1　案例一：智能温控系统

智能温控系统 [36] 在当今社会能源节约问题上起到十分重要的作用。该系统主要包含五个部分：房间温度, 控制器, 室外温度，加热器及用户。控制器根据用户的行为及室内温度来控制加热器的开关，室内温度又受到室外温度、加热器及用户行为的影响。该系统的主要目的是评估某种控制策略下房间温度的舒适度及整个系统的能耗。

6.1.1　系统建模与设计

6.1.2　系统验证分析

本实验的分布式环境为五台处理器为英特尔 (TM) i7-4790 (八核，主频 3.6G 赫兹) 组成的集群。为了评估系统的行为和算法的性能，我们验证了以下三个验证属

表 6.1: 智能温控系统的验证属性

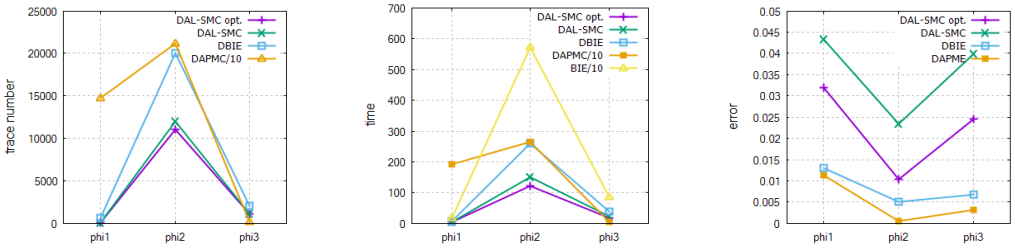
序号	(δ, c)	验证属性	验证结果
ϕ_1	$(0.05, 0.99)$	$P_{=?}(F^{\leq 48}\ energy \geq 210)$	0.1778
ϕ_2	$(0.01, 0.99)$	$P_{=?}(F^{\leq 48}\ discomfort \geq 15)$	0.4861
ϕ_3	$(0.02, 0.9)$	$P_{=?}(F^{\leq 48}\ discomfort \leq 15 \wedge energy \geq 170)$	0.4535

性如表 6.1所示。我们采用贝叶斯区间估计算法对上属三条验证属性进行评估分析以得到的验证结果。下面我们对表6.1的验证属性及验证结果进行分析：(1)ϕ_1 用来评估在 48 小时内能耗超过 210 的概率大小，我们得到的验证结果为 0.1778，表示在 48 小时内系统能耗超过 210 的可能性比较小;(2)ϕ_1 用来评估 48 小时内，系统的不舒适度大于 15 的概率，得到的验证结果为 0.4861;（3）ϕ_3 用来评估在 48 小时内，系统的不舒适度小于 15 且能耗大于 170 的概率，得到的验证结果为 0.4535。由此，可以发现本文提出的技术方法可以有效的评估异构系统的行为。

6.1.3　算法评估分析

为了评估章节 4 提出的算法效率，在本小节我们针对以上的三个验证属性，从产生的迹的数量，验证所需要的时间及验证误差三个方面对比了多种统计模型检测算法：贝叶斯区间估计算法（BIE）、分布式的贝叶斯区间估计算法（DBIE）、基于抽象和学习的分布式统计模型检测算法（DAL-SMC）、经过优化的基于抽象和学习的分布式统计模型检测算法（DAL-SMC opt）、分布式的 APMC[?] 算法（DAPMC），算法对比结果如图6.1所示。

图6.1(a)是算法产生的迹的数量对比，我们可以发现对于验证属性 ϕ_2，DAPMC 将消耗 200000 条迹，然而 DBIE 只需要 20000 条迹。DAL-SMC 和 DAL-SMC opt. 需要更少量的迹 (大约 10000 条)。算法的验证时间的对比如图 6.1(b)所示，对于验证属性 ϕ_2，BIE 需要 6000 秒，但分布式的 BIE 只需要大约 250 秒，DAL-SMC 和 DAL-SMC opt. 需要更少的时间。图6.1(c) 是算法的验证误差分析，对于验证属性 ϕ1，分布式 BIE 和 DAPMC 的误差大约为 0.013，DAL-SMC 和 DAL-SMC 的验证误差大约



(a) 产生的迹的数量　　(b) 验证时间　　(c) 验证误差

图 6.1: 智能温控系统案例算法对比

表 6.2: 机器人路径规划的验证属性

序号	(δ, c)	验证属性	验证结果
ϕ_4	$(0.01, 0.99)$	$P_{=?}(F^{\leq 100}\ robot.collision)$	0.2675
ϕ_5	$(0.05, 0.99)$	$P_{=?}(F^{\leq 100}\ energy \geq 500)$	0.5211
ϕ_6	$(0.02, 0.9)$	$P_{=?}(F^{\leq 100}\ robot.collision \wedge energy \geq 500)$	0.5296

为 0.045 和 0.032。更详细的实验数据如表6.3所示。

6.2　案例二：机器人路径规划系统

在近些年来, 机器人路径规划问题引起了学术界越来越多的人的关注。机器人路径规划问题的主要目标是要避免机器人与障碍物发生碰撞并最终到达目的地。本案例在经典的机器人路径规划的基础上加上能耗，即机器人在移动过程中会消耗能量且在不同的时刻消耗的能量也可能存在区别。本案例主要包含机器人及障碍物两个部分，最终我们需要评估机器人产生的能耗及机器人发生碰撞的概率大小。

6.2.1　系统建模与设计

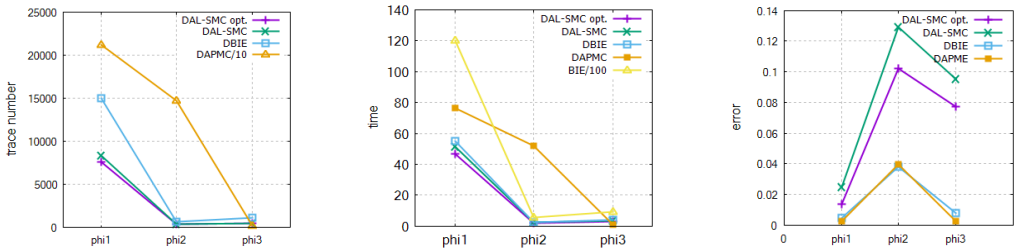
6.2.2　系统验证分析

我们也采用贝叶斯区间估计算法对机器人路径规划的三条验证属性进行评估分析以得到验证结果。下面我们对表6.2的验证属性及验证结果进行分析：(1)ϕ_4 用

来评估在 10 小时内机器人与障碍物发生碰撞的概率大小，我们得到的验证结果为 0.2675，表示在 10 小时内机器人与障碍物发生碰撞的可能性比较小;(2)ϕ_5 用来评估 10 小时内机器人消耗的能量大于 500 的概率大小，得到的验证结果为 0.5211;（3）ϕ_6 用来评估在 48 小时内，10 小时内机器人消耗的能量大于 500 的概率且不发生碰撞的概率大小，得到的验证结果为 0.5296。由此，该案例也可有效说明本文提出的技术方法可以有效的评估异构系统的行为。

6.2.3　算法评估分析

通过用多种算法对机器人路径规划案例进行评估分析，得到的算法对比结果如图 6.2所示：



(a) 产生的迹的数量　　(b) 验证时间　　(c) 验证误差

图 6.2: 机器人路径规划案例算法对比

表6.3给出了上述两个案例的算法对比的详细数据，下面我们针对验证属性 ϕ_2 来对算法的性能对比总结如下：

(i)DAPMC 的验证需要产生最多的仿真迹，相对于 DAPMC，DBIE 需要的迹的数量就少很多。此外，DAL-SMC 相对于 DBIE 又将仿真迹减少了大约一半。因此，针对产生仿真迹的数量来说，DAL-SMC 是最高效的算法。

(ii) 对于验证而言，产生仿真迹的过程消耗了主要的验证时间。在分布式算法之中，DAPMC 消耗时间最多，DAL-SMC 得益于产生较少的仿真迹而相对 DBIE 消耗的时间较少。在本实验中，我们使用 40 个核来实现分布式算法，我们发现经典 BIE 算法消耗的时间大约是分布式 BIE 算法的 25 倍，由此可以说明分布式技术在提高统计模型检测算法效率上是十分有效的。

(iii)DAPMC 和 DBIE 的验证误差较小，DAL-SMC 的验证误差相对于 DAPMC 和 DBIE 较大一些。此外, 我们发现 DAL-SMC opt. 的误差小于 DAL-SMC 的验证误差，说明我们提出的参数优化方法是有效的。

总的来说，DAL-SMC opt. 算法得益于分布式技术和抽象与学习技术，在验证过程中需要产生最少的仿真迹和消耗最少的验证时间，此外，由于使用了章节 4 提出的参数优化方法，也使得将该算法的误差控制在了一个较小的范围之内。

6.3　本章小结

本文使用两个案例来说明本文提出的方法的可行性，首先对案例进行了建模和设计，然后重点在系统的验证分析小节中展示了本方法在对异构系统进行验证分析时的有效性，并在算法评估分析小节结合两个案例，在多个方面对比了多种算法的性能，从而展示了我们本文提出的基于抽象和学习的分布式统计模型检测算法的高效性和准确性。

表 6.3: 实验结果

算法	验证属性	迹的数量	验证时间	验证误差
DAPMC	$\phi 1(0.05, 0.99)$	147000	1934.31	0.0113
	$\phi 2(0.01, 0.99)$	**211932**	**2649.15**	**0.0005**
	$\phi 3(0.02, 0.9)$	1950	38.05	0.0032
	$\phi 4(0.01, 0.99)$	211930	76.282	0.0024
	$\phi 5(0.05, 0.99)$	147550	52.206	0.0399
	$\phi 6(0.02, 0.9)$	1850	1.159	0.0026
DBIE	$\phi 1(0.05, 0.99)$	600	7.895	0.0131
	$\phi 2(0.01, 0.99)$	**20000**	**259.275**	**0.0051**
	$\phi 3(0.02, 0.9)$	2100	38.057	0.0068
	$\phi 4(0.01, 0.99)$	15000	55.281	0.0049
	$\phi 5(0.05, 0.99)$	702	2.483	0.0382
	$\phi 6(0.02, 0.9)$	1120	4.226	0.0078
DAL-SMC	$\phi 1(0.05, 0.99)$	103	5.685	0.0433
	$\phi 2(0.01, 0.99)$	**12000**	**151.785**	**0.0235**
	$\phi 3(0.02, 0.9)$	1137	22.841	0.0399
	$\phi 4(0.01, 0.99)$	8318	41.68	0.0246
	$\phi 5(0.05, 0.99)$	384	2.32	0.1295
	$\phi 6(0.02, 0.9)$	520	3.601	0.0751
DAL-SMC opt.	$\phi 1(0.05, 0.99)$	95.79	4.65	0.0319
	$\phi 2(0.01, 0.99)$	**11040**	**121.425**	**0.0103**
	$\phi 3(0.02, 0.9)$	1091	18.73	0.0246
	$\phi 4(0.01, 0.99)$	7569	36.512	0.0138
	$\phi 5(0.05, 0.99)$	349	1.81	0.102
	$\phi 6(0.02, 0.9)$	494	3.171	0.0575
BIE	$\phi 1(0.05, 0.99)$	590	175.44	0.0121
	$\phi 2(0.01, 0.99)$	**19586**	**5730.67**	**0.0047**
	$\phi 3(0.02, 0.9)$	2040	845.73	0.0063
	$\phi 4(0.01, 0.99)$	15400	1202.467	0.0044
	$\phi 5(0.05, 0.99)$	762	55.221	0.0352
	$\phi 6(0.02, 0.9)$	1150	91.98	0.0069

第七章　总结与展望

参考文献

[1] 王中杰, 谢璐璐. 信息物理融合系统研究综述 [J]. 自动化学报, 2011, 37(10): 1157 – 1166.

[2] BULYCHEV P, DAVID A, LARSEN K G, et al. Uppaal-smc: Statistical model checking for priced timed automata[J]. arXiv preprint arXiv:1207.1272, 2012.

[3] KWIATKOWSKA M, NORMAN G, PARKER D. PRISM 4.0: Verification of probabilistic real-time systems[C] // Computer aided verification. 2011: 585 – 591.

[4] FRITZSON P, ENGELSON V. Modelica - A Unified Object-Oriented Language for System Modelling and Simulation[C] // European Conference on Object-Oriented Programming. 1998: 67 – 90.

[5] ZULIANI P, PLATZER A, CLARKE E M. Bayesian statistical model checking with application to Stateflow/Simulink verification[J]. Formal Methods in System Design, 2013, 43(2): 338 – 367.

[6] GEORG H, MüLLER S C, REHTANZ C, et al. Analyzing Cyber-Physical Energy Systems:The INSPIRE Cosimulation of Power and ICT Systems Using HLA[J]. IEEE Transactions on Industrial Informatics, 2014, 10(4): 2364 – 2373.

[7] BASTIAN J, CLAUß C, WOLF S, et al. Master for co-simulation using FMI[C] //. 2011.

[8] LEGAY A, DELAHAYE B, BENSALEM S. Statistical Model Checking: An Overview[J], 2010, 6418(2) : 122 – 135.

[9] ARNOLD M, SCHIERZ T, BLOCHWITZ T, et al. FMI-for-CoSimulation[J], 2011.

[10] TRIPAKIS S. Bridging the semantic gap between heterogeneous modeling formalisms and FMI[C] // International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation. 2015 : 60 – 69.

[11] BROMAN D, BROOKS C, GREENBERG L, et al. Determinate composition of FMUs for co-simulation[C] // Proceedings of the International Conference on Embedded Software. 2013 : 1 – 12.

[12] CREMONA F, LOHSTROH M, TRIPAKIS S, et al. FIDE:an FMI integrated development environment[J], 2016 : 1759 – 1766.

[13] XXX. Improved Co-Simulation with Event Detection for Stochastic Behaviors of CPSs[C] // Computer Software and Applications Conference. 2016 : 209 – 214.

[14] LARSEN P G, FITZGERALD J, WOODCOCK J, et al. Integrated tool chain for model-based design of Cyber-Physical Systems: The INTO-CPS project[C] // International Workshop on Modelling, Analysis, and Control of Complex Cps. 2016 : 1 – 6.

[15] GROSU R, SMOLKA S A. Monte Carlo Model Checking[C] // International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems. 2005 : 271 – 286.

[16] SEN K, VISWANATHAN M, AGHA G. Statistical model checking of black-box probabilistic systems[C] // Computer Aided Verification. 2004 : 202 – 215.

[17] WALD A. Sequential tests of statistical hypotheses[J]. The Annals of Mathematical Statistics, 1945, 16(2) : 117 – 186.

[18] YOUNES H L. Verification and planning for stochastic processes with asynchronous events[R]. [S.l.] : DTIC Document, 2005.

[19] YOUNES H L, SIMMONS R G. Statistical probabilistic model checking with a focus on time-bounded properties[J]. Information and Computation, 2006, 204(9) : 1368 – 1409.

[20] HÉRAULT T, LASSAIGNE R, MAGNIETTE F, et al. Approximate probabilistic model checking[C] // Verification, Model Checking, and Abstract Interpretation. 2004 : 73 – 84.

[21] JHA S K, CLARKE E M, LANGMEAD C J, et al. A bayesian approach to model checking biological systems[C] // Computational Methods in Systems Biology. 2009 : 218 – 234.

[22] KIM Y, KIM M, KIM T H. Statistical Model Checking for Safety Critical Hybrid Systems: An Empirical Evaluation[M]. [S.l.] : Springer Berlin Heidelberg, 2012 : 162 – 177.

[23] KUMAR J A, AHMADYAN S N, VASUDEVAN S. Efficient Statistical Model Checking of Hardware Circuits With Multiple Failure Regions[J]. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2014, 33(6) : 945 – 958.

[24] BASU A, BENSALEM S, BOZGA M, et al. Statistical abstraction and model-checking of large heterogeneous systems[J]. International Journal on Software Tools for Technology Transfer, 2012, 14(1) : 53 – 72.

[25] XXX. AL-SMC: Optimizing Statistical Model Checking by Automatic Abstraction and Learning[J]. International Journal of Software and Informatics, 2016, 10 : 4:0.

[26] YOUNES H L. Ymer: A statistical model checker[C] // Computer Aided Verification. 2005 : 429 – 433.

[27] BULYCHEV P, DAVID A, LARSEN K G, et al. Checking and Distributing Statistical Model Checking[M]. [S.l.] : Springer Berlin Heidelberg, 2012 : 449 – 463.

[28] SEN K, VISWANATHAN M, AGHA G A. VESTA: A Statistical Model-checker and Analyzer for Probabilistic Systems.[C] // QEST : Vol 5. 2005 : 251 – 252.

[29] BROWN L D, CAI T T, DASGUPTA A. Interval estimation for a binomial proportion[J]. Statistical science, 2001 : 101 – 117.

[30] BOYER B, CORRE K, LEGAY A, et al. PLASMA-lab: A flexible, distributable statistical model checking library[G] // Quantitative Evaluation of Systems. [S.l.] : Springer, 2013 : 160 – 164.

[31] DORI D. Model-Based Systems Engineering with OPM and SysML[M]. [S.l.] : Springer New York, 2016.

[32] BJERKANDER M, KOBRYN C. Architecting systems with UML 2.0[J]. IEEE Software, 2003, 20(4) : 57 – 61.

[33] DAVID A, DU D, LARSEN K G, et al. Statistical Model Checking for Stochastic Hybrid Systems[J]. Electronic Proceedings in Theoretical Computer Science, 2012, 92(Proc. HSB 2012) : 187 – 199.

[34] DUNTEMAN G H. Principal components analysis : Vol 69[M]. [S.l.] : Sage, 1989.

[35] CARRASCO R C, ONCINA J. Learning stochastic regular grammars by means of a state merging method[G] // Grammatical Inference and Applications. [S.l.] : Springer, 1994 : 139 – 152.

[36] DAVID A, DU D H, LARSEN K G, et al. An evaluation framework for energy aware buildings using statistical model checking[J]. Science China Information Sciences, 2012, 55(12) : 2694 – 2707.

致　　谢

在此论文完成之际，我首先要感谢我的导师杜德慧副教授。她严肃的科学态度，严谨的治学精神，精益求精的工作作风，深深地感染和激励着我。从课题的选择到项目的最终完成，杜老师都始终给予我细心的指导和不懈的支持。三年多来，杜老师不仅在学业上给我以精心指导，同时还在思想、生活上给我以无微不至的关怀，在此谨向杜老师致以诚挚的谢意和崇高的敬意。

感谢在研究生学习期间给我诸多教诲和帮助的软件学院的各位老师和同学、以及和我一起生活两年半的室友，你们的执着、勤奋、以及对生活的态度，值得我学习。“君子和而不同”，我们正是如此！愿我们以后的人生都可以充实、快乐！

最后感谢我的家人，谢谢你们在我成长道路上支持、鼓励我，让我独立地选择自己的人生道路；同时谢谢我的女朋友，在学习、生活中对我的帮助和鼓励！

姜 凯强

二零一八年五月

攻读硕士学位期间发表论文、参与科研和获得荣誉情况

■ 已完成学术论文

[1] Cheng B, Wang X, Liu J, et al. Modana: An Integrated Framework for Modeling and Analysis of Energy-Aware CPSs[C]//Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual. IEEE, 2015, 2: 127-136. （一作）

[2] 杜德慧, 程贝, 刘静. 面向安全攸关系统中小概率事件的统计模型检测 [J]. 软件学报, 2015(2):305-320. （二作，导师一作）

[3] Cheng B, Du D. Towards a Stochastic Occurrence-Based Modeling Approach for Stochastic CPSs[C]//2014 Theoretical Aspects of Software Engineering Conference (TASE). IEEE, 2014: 162-169.（一作）

■ 参与的科研课题

[1] 信息物理融合系统的随机行为建模与验证方法研究（国家自然科学基金面上项目, 61472140）

[2] 基于统计模型检测的信息物理融合系统的验证方法研究（上海市自然科学基金项目, 14ZR1412500）

■ 获得荣誉情况

[1] 2015 年获得国家奖学金

[2] 2015 年获得华东师范大学优秀学生称号