厦門大學

Master's Thesis

Security and Timing Analysis in Discrete-Event Systems: From Verification to Synthesis

Shaowen Miao

Major: Control Engineering

June 2025

摘 要

离散事件系统 (Discrete-Event Systems, DES) 作为一类重要的动态系统数学模型, 在制造系统, 通信网络和分布式软件系统等信息物理系统的建模与分析中具有广泛应用。本论文针对DES的安全性与故障诊断问题展开深入研究, 重点解决状态估计与控制器综合等核心挑战。随着现代技术系统规模与复杂度的持续增长, 系统安全性保障已成为关键科学问题。本研究通过多维度创新视角, 包括安全性 (不透明性, 临界可观测性, 故障诊断/预测) , 模块化监督控制和定时系统等, 为DES安全分析提供了系统的理论框架和方法体系。

在基于有限状态自动机 (Finite-State Automata, FSA) 的 DES 建模框架下, 本研究首次提出两种增强型始-终状态不透明性变体：非完全强始-终状态不透明性 (Partially Strong Initial-and-Final-State Opacity, PSIFO) 和强始-终状态不透明性 (Strong Initial-and-Final-State Opacity, SIFO) 。这些创新性安全属性通过建立更严格的信息保护机制, 确保非秘密路径始终能够有效遮蔽秘密信息。为支持高效验证, 本文还设计了两种新型信息结构, 显著提升了验证效率。

针对大规模安全攸关系统中模块化 DES 的复杂度挑战, 本研究在临界可观测性理论方面取得突破。创新性地提出了不完全信息模块化 DES 监督器综合方法, 所构建的监督器不仅满足非阻塞性, 最大许可性, 可控性和常态性等基本要求, 同时保证临界可观测性, 为复杂系统的安全控制提供了有效解决方案。

为更好地适应实际工程需求, 本研究突破了传统可诊断性与可预测性的理论局限, 在 FSA, 模块化 FSA 和标签 Petri 网三类模型中首次提出弱化版本属性。通过系统分析这些弱化属性的复杂度特征, 为实际系统设计提供了灵活的选择空间, 使系统设计者能够在计算复杂度与传感器部署成本之间实现最优权衡。

在定时 DES 研究领域, 本研究聚焦于恒定时间自动机模型的分散可诊断性 (协同可诊断性) 验证问题。针对验证复杂度高的挑战, 提出了有效的降复杂度条件。特别地, 本研究首次揭示了时间信息辅助下可诊断性与协同可诊断性之间的等价关系, 为定时系统诊断提供了新的理论基础。

为进一步提高模型精确度, 本研究基于时间区间自动机框架重构了可诊断性理论体系。针对不可诊断系统, 创新性地提出通过调节可控事件时间参数来实现故障可诊断性的监督控制方法, 为实际工程应用提供了新的技术途径。

本论文在 DES 安全性与定时分析领域做出了系列原创性贡献, 包括提出 PSIFO, SIFO 和弱可预测性等新概念, 发展模块化控制方法和定时系统理论。研究成果不仅丰富了 DES的理论体系, 更为安全攸关系统的设计与实现提供了兼顾

计算效率与物理约束的实用解决方案。

关键词： 离散事件系统; 安全分析; 定时分析; 监督控制; 复杂度。

Abstract

This thesis explores the advanced security and diagnosability properties of discrete-event systems (DES), which are mathematical models widely used to describe complex dynamic systems in the field of cyber-physical systems, such as manufacturing, communication networks, and distributed software systems. The analysis of DES involves addressing challenges related to state estimation and controller synthesis. As technological systems grow larger and more complex, ensuring security has become paramount. The central focus of this thesis is to provide novel perspectives to examine these challenges, particularly through the lens of security (opacity, critical observability, fault diagnosis/prognosis), modular supervisory control, and timed systems.

First of all, two advanced variants of initial-and-final-state opacity (IFO)—partially strong initial-and-final-state opacity (PSIFO) and strong initial-and-final-state opacity (SIFO)—are introduced for DES modeled by finite-state automata (FSA). These properties impose more stringent security requirements, ensuring a non-secret-pair path always shields the secrets. Furthermore, we propose two new information structures to verify these two properties, with lower complexity compared to standard IFO.

Next, critical observability is studied within modular DES to address the complexity of large-scale, safety-critical systems composed of communicating modules. We explore the synthesis of local supervisors for modular DES with incomplete information, to realize the nonblocking, supremal, controllable, normal, and critically-observable supervisor for both local and global specifications.

Additionally, recognizing that traditional diagnosability and prognosability can be overly stringent in real-world systems, weaker versions of these properties are explored for FSA, modular FSA, and labeled Petri nets. The complexity results of deciding these weaker properties vary, offering more options for practical systems, which balance computational complexity with physical costs such as sensor requirements.

The thesis extends to timed DES, focusing on decentralized diagnosability (codiagnosability) using constant-time automata. Codiagnosability requires faults to be detected by at least one local agent within a finite number of observations. Due to the intractable complexity of verification brought by this extension, we propose a condition for complexity reduction of verification. Furthermore, an interesting equivalence between diagnosability and codiagnosability with the help of timing information is found.

Finally, diagnosability is further examined using time-interval automata, allowing a more precise model of real-world systems. Moreover, supervisory control is employed to enforce diagnosability, by regulating the timing of certain controllable events, to prevent the faults from occurring silently.

This thesis contributed to the fields of security and timing analysis in DES, introducing novel concepts such as PSIFO, SIFO, and weak prognosability, and exploring modular control and timed systems. These findings offer practical solutions for security-critical systems, balancing computational efficiency with implementation constraints.

Keywords: Discrete-Event System; Security; Timing; Supervisory Control; Complexity.

Contents

图索引

第1章　绪论

1.1　研究背景及选题意义

信息物理系统 (Cyber-Physical Systems, CPS) 将传感、计算、控制和网络集成到物理对象和基础设施中, 并将它们通过互联网实现相互连接[1]。CPS具有以下主要特点: 1) 广泛应用智能和计算设备, 支持实时决策与优化; 2) 设备间通过网络实现信息传输与任务协同; 3) 物理过程实体与逻辑计算设备高度耦合, 形成一体化的系统。随着科技的进步, CPS正逐步渗透到我们的日常生活中, 涵盖智能制造, 智能交通, 智能电网及智能家居等多个应用领域。

在智能制造领域, CPS通过将制造设备和生产线与信息系统中的数据处理及决策支持功能紧密结合, 实现实时监督、智能决策与定制化生产; 在智能交通系统中, CPS集成了物联网、传感器、通信技术以及大数据分析, 通过智能导航、路线优化、事故预警及应急响应等功能, 提升了交通系统的安全性与运行效率; 智能电网应用中, CPS将传统电网与数字系统深度融合, 实现了智能负荷调控, 故障检测与自愈以及分布式能源管理, 从而提高了电力系统的可靠性、效率和灵活性; 在智能家居中, CPS通过整合各种家居设备并对其进行统一管理, 支持设备的自动化控制、远程监督、安全警报及能源优化管理。

(a) 智能制造

(b) 智能交通

(c) 智能电网

(d) 智能家居

图 **1.1** **CPS**的实际案例

由此可以看出, CPS 是混杂异构的大规模开放系统。然而, 开发这样的系统往往是困难, 昂贵, 且容易出错的。2019年前后, 波音737 MAX 8型客机曾在不到半年的时间内, 接连发生两起重大空难, 均与传感器故障及系统误判有关, 共导致346人丧生。1996年6月4日, Ariane-5运载火箭的首次发射即以失败告终。由于代码中的整数溢出问题导致火箭偏离轨道并最终解体自毁。酿成了史上最著名、代价最高的软件漏洞之一, 损失超过3.7亿美元。奔腾浮点除错误是英特尔早期奔腾系列处理器中的一个浮点运算错误, 源自浮点除法指令的设计缺陷。该错误引发了英特尔的公关灾难, 并最终导致公司在1994年遭受了约4.75亿美元的损失。1994年4月, 丹佛国际机场首次对其新开发的自动化行李系统进行了测试。然而, 测试过程中, 行李被散落在轨道和推车下, 且推车频繁抛出行李, 导致现场一片混乱。该系统始终面临维护难题, 最终于2005年9月被停用, 机场转而重新采用传统的人工行李处理方式。

通过上述例子, 可以看出, 当前阶段 CPS 的分析与设计面临以下几个主要问题: 1) 任务描述依赖自然语言, 缺乏严格的形式化定义; 2) 解决方案主要基于工程经验, 无法充分保证其正确性; 3) 项目上线前需反复进行测试与迭代, 导致设计周期过长。这些问题凸显了引入离散事件系统[2] (Discrete-Event Systems, DES) 等形式化方法的必要性。形式化方法旨在通过统一的数理逻辑语言和模型对任务进行精确描述, 消除潜在歧义。在解决方案的生成过程中, 形式化方法希望通过程序化与算法化的设计流程, 减少甚至避免测试迭代。由于设计结果的正确性可以通过形式化验证方法进行严格保证, 这为系统的安全性提供了坚实的保障。

与受自然法则支配、接近物理层并以时间驱动、通过微分和差分方程描述的系统[3-4]不同, DES是一类状态空间离散且以事件驱动的动态系统。该类系统更适用于CPS信息层的任务描述与决策, 其动态特征表现为离散事件的异步发生。这些事件包括可控的 (如按键操作、设备开启或消息包的发送) , 以及不可控的 (如设备自发故障或数据包丢失) 。此外, 一些事件可以被传感器监测到, 而另一些则无法观测。DES的常见模型包括自动机、Petri网和正则语言等, 主要研究两个核心问题: 一类是验证问题, 即给定系统及任务规约, 如何验证该系统满足该规约; 另一类是综合问题, 即当系统不满足规约时, 如何设计控制器使其行为符合规约。

鉴于CPS的广泛应用前景、当前面临的挑战, 以及DES对CPS的适配性, 本文以“离散事件系统的安全与定时分析: 从验证到综合”为题, 重点从安全性质和定时模型两个角度, 深入探讨DES的性质验证[5]和控制器综合[6]这两类基本问题。

1.2　文献综述

1.2.1　离散事件系统的安全性

随着网络系统中信息交换的迅速增长, 信息安全正成为备受关注的话题[7-8]。不透明性 (Opacity) 是一种基于信息流的属性, 用于评估系统隐匿敏感信息免受外部入侵者的能力。作为 DES 中描述安全性的重要属性之一, 不透明性在保护共享网络基础设施上的秘密信息方面具有广泛的应用[9-12]。

不透明性的概念最初在文献 [13] 中被提出, 用于研究安全协议的验证。随后, 该概念被引入到 DES 领域, 从而衍生出满足不同安全需求的多种不透明性概念。对于以有限状态自动机 (Finite State Automata, FSA) 建模的 DES, 最早提出的基于状态的不透明性是文献 [14] 中的当前状态不透明性 (Current-State Opacity, CSO) 。而文献 [15] 的工作则将注意力转向初始状态估计, 并引入了初始状态不透明性 (Initial-State Opacity, ISO) 的概念。随后, 基于延迟状态估计, 两个相关概念, 即 K-步不透明性 (K-Step Opacity, K-SO) [16] 和无限步不透明性 (∞-Step Opacity, ∞-SO) [17] 被引入。受通信网络安全需求的启发, 文献 [18] 引入了一种新的不透明性概念, 称为始-终状态不透明性 (Initial-Final-State Opacity, IFO) 。该概念将秘密信息定义为一组状态对, 因而需要同时考虑初始状态估计和当前状态估计[19]。此外, IFO 还有其他应用场景, 如网络系统[18]、多智能体系统[20] 等。

为了提高不透明性验证的效率, 文献中提出了多种算法。这些算法已在文献 [21] 中得到了广泛的总结和分析。文献 [18] 提出了利用反向自动机的观测器来验证 ISO。与 [15] 中的算法相比, 该算法具有更低的指数复杂度。随后, 在文献 [22] 中, 提出了一种更高效的基于双向观测器的算法, 用于验证 K-SO 和 ∞-SO。值得注意的是, 文献 [23] 对 [22] 中提出的算法的复杂度进行了修正。此外, 文献 [18, 21, 24] 详细分析了前述不透明性概念之间的转换关系。在文献 [10, 25]中, 对不透明性与可检测性[26-29]等其他性质的关系进行了探讨。

近年来, 随着对安全性要求的提高, 诞生了一些具有更高安全性的不透明性概念, 统称为强不透明性, 这些概念能够更可靠地保护秘密信息。强 K 步不透明性 (Strong K-SO, K-SSO) 的概念最早在文献 [30] 中被提出。随后, 文献 [31] 引入了强无限步不透明性 (∞-SSO) 的概念, 该概念要求系统是否处于秘密状态的信息在整个观测过程中始终对入侵者不可知。文献 [32] 中定义了两种强不透明性概念, 分别是强当前状态不透明性 (SCSO) 和强初始状态不透明性 (SISO) 。SCSO (SISO)

表明, 如果一条路径通向一个秘密状态 (从一个秘密状态出发) , 则应存在另一条从未经过秘密状态的路径, 但能产生相同的观测结果。此外, 文献 [32-34] 提出了用于验证 K-SSO 和 ∞-SSO 的更高效算法。文献 [35] 探讨了 ∞-SO 与 SCSO 之间的转换, 为强不透明性与标准不透明性之间的关系提供了新的见解。强不透明性确保对于任何可能导致秘密泄露的系统行为, 始终存在另一种替代行为, 能够在不暴露任何秘密的情况下有效保护秘密信息。需要注意的是, 当前的 IFO 要求并不能保证沿着从初始状态到最终状态的路径不泄露其他秘密状态对, 这引发了本文第 3 章关于强始-终状态不透明性的工作。

另一方面, 现代技术系统正朝着更加复杂、规模更大和安全要求更高的方向发展。作为大规模 DES 的形式化控制方法, 模块化监督控制 (Modular Supervisory Control) [36-37] 有助于大型系统的管理, 并通过仅综合局部监督器来降低计算复杂度。此外, 当出现新的控制要求时, 仅需重新计算一些局部监督器。

众所周知, 语言的常态性 (Normality) 使观测者能够仅通过观测字符串来确定安全语言中字符串的归属。常态性可以看作是区分每对字符串的能力, 其中一个是安全的, 另一个不是。语言的常态性可以追溯到在文献 [38] 中的引入, 并随后在文献 [39-43] 中得到了广泛的研究。与可观性 (Observability) 不同, 常态性在并集下保持, 因此常态语言的极大常态子语言始终存在。

文献 [44] 在 (模块化) FSA 的背景下引入了临界可观测性 (Critical Observability, CO) 的概念, 即根据事件观测区分关键和非关键状态的能力。此外, CO 也在 Petri 网 [45-46] 的框架中进行研究。文献 [45] 深入研究了 (模块化) FSA 和 Petri 网的 CO 验证的计算复杂度。具体而言, 决定 CO 对于 FSA 是 NL-complete, 对于模块化 FSA 是 PSPACE-complete, 对于标签 Petri 网 (Labeled Petri Nets, LPN) , 除非关键标记有限, 否则是不可判定的。此外, 在文献 [46] 中, 使用基础标记 (Basis Marking) 解决了有界 LPN 的 CO 使能问题。

不同于常态性, CO 的定义基于状态。另一个区别是, 常态性是在语言的前缀闭包上定义的。然而, 本文发现这两个概念密切相关, 并且对于具有特殊形式的关键状态的确定性有限状态自动机 (Deterministic Finite-state Automata, DFA) 而言, 它们是等价的。因此, CO 与常态性共享一些很好的属性, 如存在极大子语言。

本文还将从故障诊断及预测的角度探讨 DES 的安全性。DES 的可诊断性 (Diagnosability) 验证是一个常见的话题, 近几十年来引起了极大关注 [2,9] 。可诊断性验证问题涉及确定是否可以在有限的观测内检测到故障。可诊断性研究最早可追溯

到文献 [47] 和 [48]。前者提出了一种基于状态的故障诊断方法, 而后者则基于故障事件形式化定义了可诊断性。此外, 文献 [48] 中提出了一种诊断器, 它是具有 N (表示正常) 和 F (表示故障) 标签的观测器, 用于验证 FSA 的可诊断性。然而, 诊断器的构造相对于系统的状态数量具有指数复杂度。随后, 在 [49] 中提出了第一个多项式时间的可诊断性验证方法, 该方法利用所研究自动机的两个相同观测系统的并行组合。此外, 可诊断性验证工具在后续研究中不断得到提升[50-51]。

然而, 可诊断性的一个潜在缺点是它要求在故障已经发生后才检测出故障。虽然及早发现故障可以减轻损失, 但仍有可能造成不可预见的危害。在 [52] 中, 引入了可预测性 (Prognosability/Predictability) 来提前预测故障的发生。具体而言, 它描述了一种场景, 即基于某些观测, 可以在故障实际发生前几步检测到故障。有趣的是, 非确定性有限状态自动机 (Nondeterministic Finite-state Automata, NFA) 的可预测性验证可以使用现有工具, 如诊断器[48] 和验证器[50]。因此, 其验证复杂度并不比可诊断性高。文献 [53-54] 研究了模块化 DES 的可诊断性和可预测性。

此外, 文献中还广泛研究了可诊断性[55-58] 和可预测性[59-62] 在 LPN 上的验证。文献 [55] 首先引入验证器网, 然后利用其可达树和覆盖树分别验证有界和无界 LPN 的可诊断性。对于有界情况, [56] 提出了扩展基础可达树, 并基于此构建验证器来验证可诊断性。此外, 决定无界 LPN 的可诊断性的问题被证明是 EXPSPACE-complete[58]。在 [57-58] 中, 讨论了一些 Petri网的子类的可诊断性的验证复杂度。

在 [59] 中, 提出了一个充分条件来验证有界和无界的部分可观 Petri 网的可预测性。在 [60] 中, 无界 LPN 的可预测性验证被简化为模型检测 (Model Checking) 问题, 表明它是EXPSPACE-complete。此外, [61] 构建了一个具有指数复杂度的预测图来验证有界和无界 LPN 的可预测性。在 [62] 中引入了一个可预测性验证器, 略微降低了有界 LPN 的可预测性验证的复杂度。

对系统的更多观测通常需要更多的传感器, 而可诊断性的要求在某些应用场景中可能过于严苛。因此文献 [63] 提出了弱可诊断性, 并用测试自动机[9] 进行弱可诊断性验证, 该方法需要指数时间。使用更通用的模型来研究相似问题对于实际应用很有意义。然而, LPN 的弱可诊断性验证仍然是一个开放问题, 构成了本文的动机之一。此外, 上述可预测性也存在要求过于严苛的问题, 本文引入了弱可预测性, 并使用 (模块化) NFA 和 LPN 研究其验证复杂度。

1.2.2 定时离散事件系统的故障诊断

文献 [64-65] 首次将先前可诊断性的工作扩展到分布式框架, 并引入了协同可诊断性的概念, 要求至少一个局部诊断代理可以诊断故障。

在 [66] 中, 可诊断性的工作被首次拓展至定时DES, 定时自动机 (Timed Automata, TA) [67] 的 (T-)可诊断性验证问题被证明是 PSPACE-complete。而文献 [68] 中的工作证明了 FSA 和 TA 的 (T-)协同可诊断性验证问题都是 PSPACE-complete。

文献 [69] 中的工作扩展了 [70] 中的在线状态估计算法, 以实现最大加自动机 (Max-Plus Automata, MPA) [71-72] 的故障诊断。在 [73] 中, 研究了无歧义MPA的 (T-)可诊断性。文献 [74] 和 [75], 分别为具有克隆性质的多项式模糊MPA, 和确定性标签 CTA 构建了观测器。在 [76] 中, 研究了幺半群上加权自动机的 T-可诊断性, 其验证方法基于系统及其正常子自动机的并发组合, 并证明确定性 MPA 的 T-可诊断性验证问题是 coNP-hard。文献 [77] 中开发了时间区间自动机 (Time-Interval Automata, TIA) 的诊断器。文献 [78] 研究了加权自动机[79] 的 (T-)协同可诊断性。

本文第 6 章研究 TA 的一个子类, 称为恒定时间自动机 (Constant Time Automata, CTA) , 其特点是只配备一个时钟, 每次事件发生都会重置时钟, 且时间约束恒定。CTA 利用变迁权重来表示状态变迁所需时间, 从而比 FSA 更准确地模拟现实系统。与 [76] 相比, 本文使用正常子自动机和故障子自动机的定时并行组合来验证协同可诊断性, 并给出了复杂度约简条件。与 [78] 相比, 本文的方法使用时间信息来修正协同可诊断性, 无需 [78] 中关于底层 FSA 的协同可诊断性的假设。

前述关于故障诊断的工作主要是不同模型的可诊断性验证。而当系统不可诊断时, 则需要引入控制方法, 使能系统的可诊断性。监督控制作为控制 DES 的形式化方法, 被广泛用于保证系统满足某些特定要求 (如安全性、活性等) [2,6] 。使用监督控制使能系统可诊断性的问题称为主动诊断 (Active Diagnosis) [80] 。

在 [81] 中, 主动诊断问题的研究被扩展到概率自动机[82] 框架。在 [83-84] 中, 控制器被假设具有观测系统静默的能力, 并针对主动诊断问题设计了更大许可的监督器。静默是一种超时机制, 用于推断系统是否处于死锁状态。文献 [85] 从博弈论的角度, 将主动诊断问题表述为Büchi博弈, 并证明了主动诊断决策是EXPTIME-complete。在 [86]中, 作者引入了全使能结构控制器, 以使能包括可诊断性等不同性质。此外, [87]基于验证器[50], 引入了一种无停止主动诊断控制策略来使能可诊断性, 并防止静默阻塞。在最新的进展中, [88]中的研究结合可控事件和可强制事

件来设计主动故障隔离监控器。在 [89] 中, 将主动诊断应用到电池管理系统。在 [90] 中, 引入了一种在线主动诊断算法, 仅构建部分的诊断自动机, 降低了算法复杂度。最近, 有研究针对自动机[91]和Petri网[92]讨论了异步可诊断性的使能。

另一种使能可诊断性的方法是向系统添加传感器, 以修改其观测。文献 [93] 通过选择一组成本最低的可观事件来使能可诊断性, 并将此问题转化为马尔可夫决策问题。在 [94] 中, 研究了逻辑和随机自动机的动态传感器激活。[95] 提出了一种用于使能可诊断性的最佳动态传感器激活策略。在 [96] 中, 推广了最大许可观测器, 用于使能多种性质, 包括可诊断性。此外, 可诊断性使能也在 Petri 网中得到广泛研究, 并采用各种方法, 如监督控制[97]、传感器选择[98-100] 和数字孪生[101]。

前面提到的工作主要关注逻辑 DES, 它缺乏对某些现实世界系统至关重要的时间信息。文献 [102] 在定时事件图框架下研究了一类时间故障的控制器综合问题。文献 [103] 中为 MPA 开发了一个监督控制框架, 从而得出了 MPA 的极大可控语言。文献 [104] 中的研究应用并扩展了 [103] 中的结果, 以综合 MPA 的时前最大许可和时后最小许可监督器。在最近的一项研究中, 可强制事件被用于具有完全观测的 TA 的监督控制。可强制事件可以抢占时间的流逝, 从而先于 (因此禁用) 不可控事件。此外, [105] 中提出了一个名为时间区间 DES 的新框架。

然而, 这些相关工作并没有针对部分观测的定时 DES 研究监督控制。本文考虑定时 DES 的时间信息, 为部分观测下的 TIA 设计主动诊断监督器。值得注意的是, 本文第 7 章是部分观测的定时 DES 的监督控制的首项研究①。

1.3　本文主要贡献和章节安排

本文内容安排如下:

第 1 章, 引言: 简要介绍DES的定义、应用领域及其在信息物理系统中的重要性。阐述随着技术系统的日益复杂和庞大, 如何保证系统的安全性以及如何在定时框架下研究系统性质已成为首要挑战。然后概述本论文的主要贡献, 特别是在系统安全性及定时系统的创新点。最后概述各章节内容安排及贡献。

第 2 章, 理论基础: 介绍DES的基本模型, 包括自动机, 模块化系统, 定时系统, 及Petri网的数学描述。介绍DES状态估计与控制器综合的基本问题与工具。

之后的五个正文章节分为两大模块, 即 DES 的安全分析及定时分析, 研究脉络均为从性质验证起, 至控制器综合, 第 3 章从不透明性的角度探讨了DES安全性, 并

① 文献 [106] 表明, 即使对于仅考虑内部 (状态) 规范的 TA, 控制器综合问题也是不可判定的。

给出相应验证方法; 第 4 章从临界可观测性的角度考虑大规模DES的安全性, 并给出控制器综合方法; 第 5 章作为过渡章节, 从故障诊断及预测的角度剖析DES安全性的理论分析与物理部署的关系, 为后续两章定时系统上的故障诊断与控制做铺垫; 第 6 章将故障诊断推广至分布式定时框架, 并研究验证算法; 第 7 章考虑了更为精确的定时模型, 提出了一种新的监督控制方法; 具体如下。

第 3 章 ([107]), 强始-终状态不透明性: 首先回顾标准的始-终状态不透明性的定义, 指出其在保密程度上的局限性。随后引出两类增强的始-终状态不透明性, 给出对应验证算法, 并详细分析其复杂度。最后证明从已有不透明性到本文提出的不透明性的归约, 阐述其适用性。

第 4 章 ([108-109]), 模块化系统的临界可观测性使能: 首先介绍临界可观测性的概念, 并讨论常态性与临界可观测性的关系。随后提出计算极大可控、常态及临界可观子语言的算法。最后讨论针对局部及全局规范的模块化控制器综合。

第 5 章 ([110-111]), 弱可诊断性和弱可预测性的问题可判定性: 提出弱版本的可诊断性与可预测性, 改善了标准定义的缺点。基于FSA, 模块化FSA及Petri网三类模型, 详细讨论了性质的验证复杂度及可判定性。

第 6 章 ([112]), 恒定时间自动机的协同可诊断性分析: 首先讨论了与有限状态自动机相比, 在恒定时间自动机框架下研究可诊断性的优势与差异。定义恒定时间自动机的协同可诊断性, 强调其在分布式环境下的重要作用, 即利用多个代理协同诊断故障。随后给出相应验证算法, 并详细讨论其复杂度, 给出了一种复杂度约简条件。最后在定时框架下, 证明了协同可诊断性与集中可诊断性的等价性。

第 7 章 ([113]), 时间区间自动机的主动诊断: 为时间区间自动机定义可诊断性, 并构造相应验证器。提出了一种利用时间信息进行主动诊断的控制策略。最后以智能家居系统案例探讨所提方法的潜在应用场景。

第8章, 总结与展望: 总结本文主要贡献, 并探讨进一步可拓展方向。

第2章　预备知识

本章节介绍本文所涉及的预备知识, 包括自动机理论 (有限状态自动机[2]、恒定时间自动机[75]、时间区间自动机[114-115]) , Petri网理论[116], 监督控制理论[6], 复杂度理论[117], 以及一些专业术语和符号。

2.1　自动机理论

2.1.1　有限状态自动机

对于一个集合 A, 符号 $|A|$ 表示 A 的基数 (元素个数) 。字母表 Σ 是一个有限的非空标签集合。字符串是由字母表 Σ 中的标签组成的序列。记 Σ^* 为所有 Σ 上有限长的字符串的集合, 包括空字符串 ε。对于字符串 $w \in \Sigma^*$, 用符号 $|w|$ 表示 w 的长度, 定义$|\varepsilon| = 0$。对于两个字符串 $w_1, w_2 \in \Sigma^*$, 记 $w_1 \cdot w_2$ (或简记 $w_1 w_2$) 为 w_1 和 w_2 的**串接**。一个 Σ 上的语言是 Σ^* 的一个子集, 即 $L \subseteq \Sigma^*$。用 $\overline{L} = \{w \in \Sigma^* \mid \exists v \in \Sigma^* : wv \in L\}$ 表示 L 的前缀闭包。如果$L = \overline{L}$, 则称语言L是前缀闭合的。字符串 $w_1 \in L$ 的**后语言** (或称为左商) 是集合 $L/w_1 = \{w_2 \in \Sigma^* \mid w_1 w_2 \in L\}$。

投影$R : \Sigma^* \to \Gamma^*$, 其中 $\Gamma \subseteq \Sigma$, 是一个态射, 其定义为：当 $e \in \Sigma \setminus \Gamma$ 时, $R(e) = \varepsilon$; 当 $e \in \Gamma$ 时, $R(e) = e$。对于字符串 $e_1 e_2 \cdots e_n \in \Sigma^*$, 操作 R 的作用是移除不属于 Γ 的事件, 即 $R(e_1 e_2 \cdots e_n) = R(e_1)R(e_2) \cdots R(e_n)$。对应的逆映射 $R^{-1} : \Gamma^* \to 2^{\Sigma^*}$ 定义为 $R^{-1}(t) = \{s \in \Sigma^* \mid R(s) = t\}$。这些定义可以用常规方法扩展定义域到语言上。

非确定性有限状态自动机 (Nondeterministic Finite Automata, NFA) 是一个五元组 $G = (Q, \Sigma, \delta, Q_i, Q_m)$, 其中 Q 是一个有限状态集合, Σ 是一个字母表, $Q_i \subseteq Q$ 是初始状态集合, $Q_m \subseteq Q$ 是标记状态的集合, $\delta : Q \times \Sigma \to 2^Q$ 是变迁函数, 并且可以按照通常的方式扩展定义域到 $2^Q \times \Sigma^*$。如果对于每个状态 $q \in Q$ 和每个事件 $e \in \Sigma$, $|\delta(q, e)| \leq 1$, 且 q_0 是唯一的初始状态, 则自动机 G 是**确定性的** (Deterministic Finite Automata, DFA) , 此时变迁函数可视为 $\delta : Q \times \Sigma \to Q$。自动机 G 的**生成语言**是集合 $L(G) = \{w \in \Sigma^* \mid \delta(q_0, w) \in Q\}$, 自动机 G 的**标记语言**是集合 $L_m(G) = \{w \in \Sigma^* \mid \delta(q_0, w) \in Q_m\}$。根据定义, $L_m(G) \subseteq L(G)$, 且 $L(G)$ 是前缀闭合的。如果 $L_m(G) = L(G)$, 则称 G 为**非阻塞的** (Nonblocking) 。

对于任意状态子集 $Q' \subseteq Q$, 其不可观到达定义为 $UR(Q') = \{q \in Q \mid \exists q' \in$

$Q', \exists s \in \Sigma_{uo}^* : q \in \delta(q', s)\}$。给定一个 NFA G, 其**观测器**[2,118]定义为：

$$Obs(G) = (X, \Sigma_o, f, x_i) = Ac(2^Q, \Sigma_o, f, UR(Q_i)), \tag{2-1}$$

其中 $Ac(\cdot)$ 表示可达部分, 并且 $f : X \times \Sigma_o \to X$ 是确定性变迁函数, 定义为 $f(x, e) = UR(\{q \in Q \mid \exists q' \in x : q \in \delta(q', e)\})$。如果 $UR(\{q \in Q \mid \exists q' \in x : q \in \delta(q', e)\}) = \emptyset$, 则 $f(x, e)$ 未被定义。观测器描绘了系统的可观行为, 允许进行当前状态估计。

给定一个 NFA $G = (Q, \Sigma, \delta, Q_i)$, **增强自动机**[119]被提出用于初始状态估计：

$$G^{aug} = (Q^{aug}, \Sigma, \delta^{aug}, Q_i^{aug}) = Ac(Q_i \times Q, \Sigma, \delta^{aug}, Q_i^{aug}),$$

其中 $Q_i^{aug} = \{(q, q) : q \in Q_i\}$, 且变迁函数 $\delta^{aug} : Q^{aug} \times \Sigma \to 2^{Q^{aug}}$ 定义为 $\delta^{aug}((q_i, q_f), e) = \{(q_i, q_f') : q_f' \in \delta(q_f, e)\}$。注意到 G^{aug} 里的每个状态 $(q_i, q_f) \in Q^{aug}$ 都是一个状态对, 表示当前状态 q_f 从初始状态 q_i 可达。系统 G 的初始状态观测器 $Obs(G^{aug})$ 是增强自动机 G^{aug} 的观测器, 构造如下：

$$Obs(G^{aug}) = (X^{aug}, \Sigma_o, f^{aug}, x_i^{aug}) = Ac(2^{Q^{aug}}, \Sigma_o, f^{aug}, UR^{aug}(Q_i^{aug})), \tag{2-2}$$

其中增强不可观到达 $UR^{aug}(Q')$ 为任意增强状态子集 $Q' \subseteq Q^{aug}$ 按如下定义：$UR^{aug}(Q') = \{(q_i, q_f) \in Q^{aug} \mid \exists (q_i, q_f') \in Q', \exists s \in \Sigma_{uo}^* : q_f \in \delta(q_f', s)\}$, 且 f^{aug} 定义方式同 $Obs(G)$ 中的 f。

2.1.2 恒定时间自动机

设 $\mathbb{R}_{\geq 0}$ 表示非负实数集。**恒定时间自动机** (Constant-Time Automata, CTA) 是一个五元组 $G = (Q, \Sigma, Q_i, \Delta, \mu)$, 其中 $\Delta \subseteq Q \times \Sigma \times Q$ 是变迁关系, 可拓展为 $\Delta^* \subseteq Q \times \Sigma^* \times Q$; 时间标签函数 $\mu : \Delta \to \mathbb{R}_{\geq 0}$ 为每个变迁分配一个 $\mathbb{R}_{\geq 0}$ 中的值。

CTA G 中长度为 k 的**路径**定义为状态变迁的序列：

$$\pi = (q_0, e_1, q_1)(q_1, e_2, q_2) \cdots (q_{k-1}, e_k, q_k),$$

其中 $q_i \in Q$, $i = 0, 1, \ldots, k$; $e_i \in \Sigma$ 及 $(q_{i-1}, e_i, q_i) \in \Delta$, $i = 1, 2, \ldots, k$。当路径 π 的起始状态 q_0 与终止状态 q_k 重合, 且路径中没有其他状态被重复, 称 π 为一个 (简单) **循环**。当 $q_0 \in Q_i$, 由 π 生成的**定时字符串** $\mathcal{TW}(\pi) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ 定义为一个由 (事件, 时间) 序偶组成的序列：$\mathcal{TW}(\pi) = (e_1, \tau_1)(e_2, \tau_2) \cdots (e_k, \tau_k)$, 其中

$\tau_i = \sum_{j=1}^{i} \mu(q_{i-1}, e_i, q_i)$, $i = 1, 2, \ldots, k$。定时字符串 $\mathcal{TW}(\pi)$ 的每个事件的发生时刻由相应序偶的第二个元素表示。由字符串 $w \in \Sigma^*$ 标记、从状态 q_1 到状态 q_2 的所有路径的集合记为 $q_1 \overset{w}{\rightsquigarrow} q_2$。当 $w = \varepsilon$ 且 $q_1 \neq q_2$ 时, $q_1 \overset{w}{\rightsquigarrow} q_2$ 是一个空集。当不关心标签时, 简记为 $q_1 \rightsquigarrow q_2$。对于任意逻辑字符串 $w = e_1 e_2 \cdots e_k$, 定义

$$\mu(q_0, w, q_n) = \bigcup_{\pi \in q_0 \overset{w}{\rightsquigarrow} q_n} \big(\mu(q_0, e_1, q_1) + \mu(q_1, e_2, q_2) + \cdots + \mu(q_{n-1}, e_n, q_n)\big).$$

对于任意的定时字符串 $w = (e_1, \tau_1)(e_2, \tau_2) \cdots (e_k, \tau_k) \in (E \times \mathbb{R}_{\geq 0})^*$, $\ell(w) = e_1 e_2 \cdots e_k \in E^*$ 表示忽略时间信息后的逻辑字符串, 并用 $\ell_{\text{last}}(w) = e_k$ 表示 w 中的最后一个事件。此外, $\mathbf{T}_{\text{last}}(w)$ 表示 w 中最后一个事件 e_k 的发生时间, 即 $\mathbf{T}_{\text{last}}(w) = \tau_k$。符号 $e \in s$ 表示事件 $e \in \Sigma$ 发生在字符串 $s \in \Sigma^*$ 中。

CTA G 生成的定时语言定义为：

$$L(G) = \{w \in (\Sigma \times \mathbb{R}_{\geq 0})^* \mid \exists q_0 \in Q_i, \exists q \in Q, \exists \pi \in q_0 \rightsquigarrow q : \mathcal{TW}(\pi) = w\}.$$

定时语言 $L(G)$ 的底层逻辑语言记为 $\ell(L(G)) = \{\ell(\omega) \mid \omega \in L(G)\}$。将 $\Sigma \times \mathbb{R}_{\geq 0}$ 视为一个定时字母表, 则在逻辑字母表 Σ 上定义的运算 (如 Kleene 闭包、串接、 (逆) 投影等) 可以直接扩展到 $\Sigma \times \mathbb{R}_{\geq 0}$。

2.1.3　时间区间自动机

一个 (实数) 区间是所有位于两个固定端点之间且没有间隙的实数的集合。具体而言, 对于任意 $a, b \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$ 且 $a \leq b$, 有 $(a, b) = \{x \in \mathbb{R}_{\geq 0} \mid a < x < b\}$, $[a, b] = \{x \in \mathbb{R}_{\geq 0} \mid a \leq x \leq b\}$, 以及 $(a, b] = \{x \in \mathbb{R}_{\geq 0} \mid a < x \leq b\}$。为了方便起见, 圆括号可表示开区间或元组, 方括号可表示闭区间或矩阵, 具体含义取决于上下文。两个有限非空实数子集 (例如区间) A 和 B 的加法定义为 $A + B = \{a + b \in \mathbb{R}_{\geq 0} \mid a \in A, b \in B\}$[120]。

时间区间自动机 (Time-Interval Automata, TIA) 是五元组 $G = (Q, \Sigma, Q_i, \Delta, \mu)$, 其中$\mu : \Delta \to 2^{\mathbb{R}_{\geq 0}} \setminus \emptyset$ 为每个变迁赋予一个时间约束 (区间或区间的并集) 。在 TIA G 中, 长度为 n 的**运行**是从初始状态开始的一系列状态变迁, 即：

$$\rho = q_0 \xrightarrow{\sigma_1/\tau_1} q_1 \xrightarrow{\sigma_2/\tau_2} \cdots \xrightarrow{\sigma_n/\tau_n} q_n,$$

其中 $q_0 \in Q_i$, $q_i \in Q$, $(q_{i-1}, \sigma_i, q_i) \in \Delta$, 且 $\tau_i \in \mu(q_{i-1}, \sigma_i, q_i)$, $i = 1, \ldots, n$。稍微

滥用符号, 对于运行上述 ρ, 其对应的定时字符串 $\mathcal{TW}(\rho) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ 定义为：$\mathcal{TW}(\rho) = (\sigma_1, t_1)(\sigma_2, t_2)\cdots(\sigma_n, t_n)$, 其中 $t_i = \sum_{k=1}^{i} \tau_k$, $i = 1, \ldots, n$。特别地, 令 $\mathcal{TW}(q_0) = \varepsilon$。在本文中, 如无特别说明, TIA的符号使用均与CTA相同。

2.2 Petri网理论

设 $\mathbb{N}$ 表示自然数集。一个 Petri 网是一个四元组 $N = (P, T, Pre, Post)$, 其中 P 是 m 个库所的集合, T 是 n 个变迁的集合, 满足 $P \cup T \neq \emptyset$ 和 $P \cap T = \emptyset$, 而 $Pre\colon P \times T \to \mathbb{N}$ 和 $Post\colon P \times T \to \mathbb{N}$ 是分别表示有向弧从库所到变迁和从变迁到库所的前向和后向发生函数。一个**标记**是一个映射 $M\colon P \to \mathbb{N}$, 为每个库所分配令牌的数量。一个 Petri 网系统 (N, M_0) 包括一个 Petri 网 N 和一个初始标记 M_0。如果对于每个库所 $p \in P$, 都有 $M(p) \geq Pre(p, t)$, 则称变迁 t 在标记 M 中是**可用的**。在 M 中启用变迁 t 的**发射**会导向标记 M', 其中对于每个 $p \in P$, 有$M'(p) = M(p) - Pre(p, t) + Post(p, t)$。

符号 $M \xrightarrow{\sigma}$ 表示序列 $\sigma \in T^*$ 在标记 M 中是可用的, 而 $M \xrightarrow{\sigma} M'$ 表示发射序列 σ 到达标记 M'。设 $L(N, M_0) = \{\sigma \in T^* \mid M_0 \xrightarrow{\sigma}\}$ 表示所有可以从初始标记 M_0 发射出的变迁序列的集合。如果存在一个变迁序列 $\sigma \in T^*$ 使得 $M_0 \xrightarrow{\sigma} M$, 则称标记 M 在 (N, M_0) 中是可达的。

标签Petri网 (Labeled Petri Nets, LPN) 系统是一个四元组 $G = (N, M_0, \Sigma, \ell)$, 其中 (N, M_0) 是一个 Petri 网系统, Σ 是一个字母表, 而 $\ell\colon T \to \Sigma \cup \{\varepsilon\}$ 是一个标签函数, 为变迁分配标签。标签函数可以扩展到 $\ell\colon T^* \to \Sigma^*$, 定义为 $\ell(\sigma t) = \ell(\sigma)\ell(t)$, 其中 $\sigma \in T^*$ 和 $t \in T$。如果 $\ell(t) \in \Sigma$, 则称变迁 t 是可观的, 否则为不可观的。由 G 生成的语言是集合 $L(G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in L(N, M_0)\}$。

2.3 监督控制理论

2.3.1 经典监督控制

对于一个部分可观和部分可控的系统 G, 字母表 Σ 被划分为以下几个集合：**可观**事件 Σ_o, **不可观**事件 $\Sigma_{uo} = \Sigma \setminus \Sigma_o$, **可控**事件 Σ_c, 以及**不可控**事件 $\Sigma_{uc} = \Sigma \setminus \Sigma_c$。

设 $\Gamma = \{\gamma \subseteq \Sigma \mid \Sigma_{uc} \subseteq \gamma\}$ 为控制模式的集合, 且 $P : \Sigma^* \to \Sigma_o^*$ 为 G 的投影。对于给定的 G 和 S, 关于 Γ 的**监督器** S 定义为 $S : P(L(G)) \to \Gamma$。**闭环系统**用 S/G 表示, 其生成的语言定义为：$\varepsilon \in L(S/G)$; $s \in L(S/G) \wedge s\sigma \in L(G) \wedge \sigma \in S(P(s))$

当且仅当 $s\sigma \in L(S/G)$。根据可观行为 $P(s)$, 监督器 S 禁用 $\Sigma_c \setminus S(P(s))$ 中的事件。由 S/G 标记的语言定义为 $L_m(S/G) = L(S/G) \cap L_m(G)$。如果闭环系统 S/G 是非阻塞的, 即 $\overline{L_m(S/G)} = L(S/G)$, 则监督器 S 被称为非阻塞的。

语言 $K \subseteq L(G)$ 被称为相对于 $L(G)$ 和 Σ_{uc} 是**可控的**, 如果 $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$。语言 K 被称为相对于 $L(G)$、$P : \Sigma^* \to \Sigma_o^*$ 和 Σ_c 是**可观的**, 如果对于所有 $s \in \overline{K}$ 和 $\sigma \in \Sigma_c$, 若 $s\sigma \notin \overline{K}$ 且 $s\sigma \in L(G)$, 则 $P^{-1}[P(s)]\sigma \cap \overline{K} = \emptyset$。语言 K 被称为相对于 $L(G)$ 和 P 是**常态的**, 如果 $\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$。

2.3.2　模块化监督控制

语言 $L_i \subseteq \Sigma_i^*$ 的并行组合是 $\|_{i=1}^n L_i = \bigcap_{i=1}^n P_i^{-1}(L_i)$, 其中 $P_i : (\bigcup_{i=1}^n \Sigma_i)^* \to \Sigma_i^*$, $i = 1, \ldots, n$。对于模块 G_i, $L(\|_{i=1}^n G_i) = \|_{i=1}^n L(G_i)$ 和 $L_m(\|_{i=1}^n G_i) = \|_{i=1}^n L_m(G_i)$ 成立。语言 L_i 被称为**非冲突的** (Nonconflicting) , 若 $\overline{\|_{i=1}^n L_i} = \|_{i=1}^n \overline{L_i}$。

模块化系统 $G = \|_{i=1}^n G_i$ 是 $n \geq 2$ 个定义在字母表 Σ_i 上的组件 G_i, $i = 1, \ldots, n$ 的并行组合, 其中局部 (组件) 行为是 $L_i = L(G_i)$, 全局 (整体) 行为是 $L = \|_{i=1}^n L_i$。模块化控制问题的目标是针对如下形式的规范 K :

1) 局部规范 $K_i \subseteq L_i$ 的并行组合, 即 $K = \|_{i=1}^n K_i$, 或

2) 全局规范 $K \subseteq \|_{i=1}^n L_i$[①],

综合局部监督器 S_i, 使得

$$\|_{i=1}^n L_m(S_i/G_i) = L_m(S/\|_{i=1}^n G_i),$$

其中 S 是规范 K 和全局语言 L 的非阻塞最大许可的监督器。

模块化系统中使用的投影符号总结如图 2.1 所示, 其中局部可观事件的集合表示为 $\Sigma_{i,o} = \Sigma_i \cap \Sigma_o$。类似的符号也适用于其他字母表的交集。

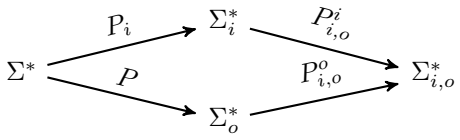

图 **2.1** 模块化系统中的投影符号

定义 2.1 (改良观测一致性 (Modified Observation Consistency, MOC) [43]). 一个前缀闭合的语言 $L \subseteq \Sigma^*$ 被称为相对于 P_i、P 和 $P_{i,o}^i$ 是改良观测一致的 (Modified

① 全局规范 $K \subseteq \|_{i=1}^n L_i$ 是对于 $\|_{i=1}^n L_i$ 可分解的, 如果 $K = \|_{i=1}^n K_i$, 其中 $K_i \subseteq L_i, i = 1, \ldots, n$。

Observation Consistent, MOC) , 如果对于任意满足 $P_{i,o}^{i}(P_i(s)) = P_{i,o}^{i}(t')$ 的字符串 $s \in L$ 和 $t' \in P_i(L)$, 存在一个字符串 $s' \in L$ 使得 $P(s) = P(s')$ 和 $P_i(s') = t'$。

如图 2.2 所示, 在局部系统中与全局字符串 s 局部观测一致的字符串 (记作 t') , 必须存在一个与其局部等价的全局字符串 s', 使得 s' 在全局上与 s 观测一致。

$$\begin{array}{ccccc} \forall s \in L & \xrightarrow{P} & P(s) = P(s') & \xleftarrow{P} & \exists s' \in L.\, P_i(s') = t' \\ \downarrow{\scriptstyle P_i} & & & & \downarrow{\scriptstyle P_i} \\ P_i(s) & \xrightarrow{P_{i,o}^{i}} & P_{i,o}^{i}(P_i(s)) = P_{i,o}^{i}(t') & \xleftarrow{P_{i,o}^{i}} & \forall t' \in P_i(L) \end{array}$$

图 **2.2 MOC** 的图解

一个模块化系统 $G = \|_{i=1}^{n} G_i$, 其公有事件集定义为 $\Sigma_s = \bigcup_{i \neq j}(\Sigma_i \cap \Sigma_j)$。本文假设任何局部可观事件如果可以在其他组件中发生, 那么其他组件也能观测到该事件, 即 $\Sigma_{i,o} \cap \Sigma_j = \Sigma_i \cap \Sigma_{j,o} = \Sigma_{i,o} \cap \Sigma_{j,o}$.

值得注意的是, 验证 MOC 是一个 PSPACE-hard 问题[42], 且其可判定性仍然是一个未解决的问题。因此, 在 [43] 中, 提出了一种易于验证的条件, 即所有公有事件都是可观的, 这一条件蕴含 MOC。

引理 2.1 ([43]中引理 5)**.** 给定一个由前缀闭合语言 $L = \|_{i=1}^{n} L_i$ 组成的模块化系统。如果 $\Sigma_s \subseteq \Sigma_o$, 则 L 相对于 P_i、P 和 $P_{i,o}^{i}$, $i = 1, \ldots, n$是 MOC。

2.4 复杂度理论

一个 **(决策) 问题**是一个是非问题, 如果存在算法可以解决它, 则称该问题是**可判定的** (Decidable) 。在复杂度理论中, 可判定的问题根据解决它们所需的时间或空间被分为不同类别。符号 NP、PSPACE 和 EXPSPACE 分别表示由非确定性多项式时间、确定性多项式空间和确定性指数空间算法解决的问题的类。一个问题被称为 NP-complete (或 PSPACE-complete、EXPSPACE-complete) , 如果：1) 它属于 NP (或 PSPACE、EXPSPACE) , 并且 2) 每个 NP (或 PSPACE、EXPSPACE) 中的问题都可以在确定性多项式时间内归约到它。条件 1) 称为**成员性** (Membership) , 条件 2) 称为**难度** (Hardness) 。一个问题属于 coNP 当且仅当它的补集在 NP 中。众所周知, NP $\subseteq$ PSPACE, 但这一包含是否严格仍然是一个悬而未决的问题。根据空间层次定理[121], 有 PSPACE $\subsetneq$ EXPSPACE。人们普遍认为, coNP-complete或PSPACE-complete问题不存在多项式时间算法。

第3章　强始-终状态不透明性

3.1　非完全强始-终状态不透明性

本节首先给出NFA的**非完全强始-终状态不透明性** (Partially Strong Initial-and-Final-State Opacity, PSIFO) 形式化定义。基于并发组合 (Concurrent Composition) 技术[32,122-125], 提出了两种验证PSIFO的方法。

秘密 (或**非秘密**) **状态对**的集合定义为 $Q_p^s \subseteq Q_i \times Q$ (或 $Q_p^{ns} \subseteq Q_i \times Q$) 。此外, 非秘密状态对集合中的初始 (或最终) 状态集合定义为 $Q_i^{ns} = \{q_i^{ns} \in Q_i \mid \exists (q_i^{ns}, q_f^{ns}) \in Q_p^{ns}\}$ (或 $Q_f^{ns} = \{q_f^{ns} \in Q \mid \exists (q_i^{ns}, q_f^{ns}) \in Q_p^{ns}\}$) 。文献 [18] 中定义的标准 IFO 的概念如下。

定义 3.1 (IFO)**.** 给定一个 NFA $G = (Q, \Sigma, \delta, Q_i)$、一个投影 P、一组秘密状态对 Q_p^s 和一组非秘密状态对 Q_p^{ns}, 如果对于所有 $(q_i, q_f) \in Q_p^s$ 和所有 $t \in L(G, q_i)$ 满足 $q_f \in \delta(q_i, t)$, 存在一个状态对 $(q_i', q_f') \in Q_p^{ns}$ 和一个字符串 $t' \in L(G, q_i')$ 使得 $q_f' \in \delta(q_i', t')$ 且 $P(t') = P(t)$, 则称 G 在 Q_p^s、Q_p^{ns} 和 P 上是始-终状态不透明的。

3.1.1　非完全强始-终状态不透明性的概念

首先引入如下非秘密对路径 (Non-Secret Pair Path, NSPP) 的概念。

定义 3.2 (NSPP)**.** 在NFA G 中, 路径 $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$ 如果对于 $j = 0, 1, \ldots, n$, 都有 $(q_0, q_j) \in Q_p^{ns}$, 则被称为 NSPP。

定义 3.3 (PSIFO)**.** 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$, 一个投影 P, 一组秘密状态对 Q_p^s, 和一组非秘密状态对 Q_p^{ns}, 若对于所有 $(q_i, q_f) \in Q_p^s$ 和所有 $t \in L(G, q_i)$ 使得 $q_f \in \delta(q_i, t)$, 存在一条NSPP $q_i' \xrightarrow{t'} q_f'$ 使得 $P(t') = P(t)$, 则称 G (对于 Q_p^s、Q_p^{ns} 和 P) 是非完全强始-终状态不透明的 (Partially Strongly Initial-and-Final-State Opaque, PSIFO) 。

例 3.1. 考虑如图 3.1 所示的自动机 G 及其增强自动机 G^{aug}, 其中秘密状态对集合为 $Q_p^s = \{(1,5), (2,1), (2,6)\}$。对于 Q_p^s, 有 $t_1 = uab^*$, 且 $5 \in \delta(1, uab^*)$; $t_2 = a$, 且 $1 \in \delta(2, a)$; 以及 $t_3 = aab^*$, 且 $6 \in \delta(2, aab^*)$。如果 $Q_p^{ns} = \{(2,5), (2,7)\}$, 则有 $t_1' = ab^*$, 且 $7 \in \delta(2, ab^*)$; $t_2' = a$, 且 $7 \in \delta(2, a)$; 以及 $t_3' = auab^*$, 且 $5 \in \delta(2, auab^*)$。根据定义 3.1, G 是IFO, 因为与秘密状态对集合对应的观测 (即 ab^* 和 aab^*) 等于非

秘密状态对集合对应的观测 (即 ab^* 和 aab^*) 。然而, 根据定义 3.3, G 不是PSIFO。这是因为, 对于秘密状态对 $(1,5)$ 和观测 ab^*, 没有对应的NSPP具有观测 ab^*。类似地, 对于秘密状态对 $(2,1)$ 和观测 a, 以及秘密状态对 $(2,6)$ 和观测 aab^*, 也没有对应的NSPP。

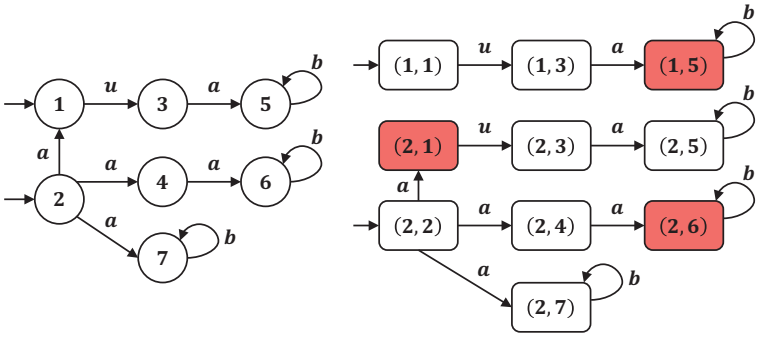

图 **3.1** 自动机 G (左) 和 G^{aug} (右) **,** 其中 $\Sigma_o = \{a, b\}$ 且 $\Sigma_{uo} = \{u\}$

由于入侵者知道系统的可观行为和完整结构, 有时即使存在一条路径可以保护目标秘密, 这条路径也可能无意中泄露其他秘密, 如例 3.1 所描述。因此, 更高的保密要求是必要的, 这引出了定义 3.3 的PSIFO。需要注意的是, PSIFO 蕴含 IFO, 但 IFO 不一定蕴含 PSIFO。

3.1.2 非完全强始-终状态不透明性的验证

本小节将验证过程分为两种情况: 一种是非秘密状态对集合 Q_p^{ns} 为一般集合, 另一种是非秘密状态对集合 Q_p^{ns} 可以表示为笛卡尔积[①]。这种划分使得适当的验证技术可以分别处理每种情况。后一种情况构成了一种特殊情况, 允许简化验证过程。这种划分并不适用于秘密状态对集合 Q_p^s, 因为验证结构并不会受此影响。

在NFA G 中, 路径 $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$ 可以直观地扩展为其在 G^{aug} 中对应的路径 $(q_0, q_0) \xrightarrow{e_1} (q_0, q_1) \xrightarrow{e_2} \cdots \xrightarrow{e_n} (q_0, q_n)$。

定义 3.4. 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$ 和一组非秘密状态对 Q_p^{ns}, 增强自动机 G^{aug} 的非秘密对子自动机, 记作

$$G_{nsp}^{aug} = (Q_{nsp}^{aug}, \Sigma, \delta_{nsp}^{aug}, Q_{nsp,i}^{aug}),$$

由 G^{aug} 推导: 首先仅保留 Q_p^{ns} 中存在的非秘密状态对, 然后应用 $Ac(\cdot)$ 操作。

① 如果一个集合 C 可以表示为 (二重) 笛卡尔积, 则存在两个集合 A 和 B, 使得 C 等于集合 A 和 B 的笛卡尔积, 即 $C = A \times B = \{(a, b) \mid \exists a \in A, \exists b \in B\}$。

G_{nsp}^{aug}的定义受到文献 [32] 中非秘密子自动机G_{dss}的启发。该子自动机的构造方法是: 首先**删除**G中所有**秘密状态**, 再取其可达部分。非秘密状态对子自动机G_{nsp}^{aug} 的观测器记为 $Obs(G_{nsp}^{aug}) = (X_{nsp}^{aug}, E_o, \delta_{nsp}^{aug}, x_{nsp,i}^{aug})$, 其具体定义遵循公式 (2-2)。

定义 3.5 (并发组合)**.** 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$ 和一组非秘密状态对 Q_p^{ns}, $G^{aug} = (Q^{aug}, \Sigma, \delta^{aug}, Q_i^{aug})$ 和 $Obs(G_{nsp}^{aug}) = (X_{nsp}^{aug}, \Sigma_o, \delta_{nsp}^{aug}, x_{nsp,i}^{aug})$ 的 **并发组合**为

$$\begin{aligned} CC(G^{aug}, Obs(G_{nsp}^{aug})) &= (Q_{cc}, \Sigma_{cc}, \delta_{cc}, Q_{cc,i}) \\ &= Ac(Q^{aug} \times X_{nsp}^{aug}, \Sigma_{cc}, \delta_{cc}, Q_i^{aug} \times \{x_{nsp,i}^{aug}\}), \end{aligned}$$

其中

- $Q_{cc} \subseteq Q^{aug} \times X_{nsp}^{aug}$ 是状态集合, 其中每个状态是一个序偶, 包含一个状态对 (左侧组件) 和一组状态对 (右侧组件) ;
- $\Sigma_{cc} = \{(e, e) : e \in \Sigma_o\} \cup \{(e, \epsilon) : e \in \Sigma_{uo}\}$ 是事件对的集合;
- $Q_{cc,i} = Q_i^{aug} \times \{x_{nsp,i}^{aug}\}$ 是初始状态集;
- $\delta_{cc} : Q_{cc} \times \Sigma_{cc} \to 2^{Q_{cc}}$ 是变迁函数, 对于任意状态 $((q_i, q_f), x_{nsp}^{aug}) \in Q_{cc}$ 和任意事件 $e \in \Sigma$, 定义如下:
  - 如果 $e \in \Sigma_o$, 则
    - 当 $x_{nsp}^{aug} \neq \emptyset$ 时,

$$\begin{aligned} &\delta_{cc}(((q_i, q_f), x_{nsp}^{aug}), (e, e)) = \{((q_i, q_f'), x'^{aug}_{nsp}) : q_f' \in \delta(q_f, e), \\ &x'^{aug}_{nsp} = \delta_{nsp}^{aug}(x_{nsp}^{aug}, e), \text{ 如果 } \delta_{nsp}^{aug}(x_{nsp}^{aug}, e) \text{ 已定义 , 否则 } x'^{aug}_{nsp} = \emptyset\}; \end{aligned}$$

    - 当 $x_{nsp}^{aug} = \emptyset$ 时, $\delta_{cc}(((q_i, q_f), \emptyset), (e, e)) = \{((q_i, q_f'), \emptyset) : q_f' \in \delta(q_f, e)\}$;
  - 如果 $e \in \Sigma_{uo}$, 则

$$\delta_{cc}(((q_i, q_f), x_{nsp}^{aug}), (e, \epsilon)) = \{((q_i, q_f'), x_{nsp}^{aug}) : q_f' \in \delta(q_f, e)\}.$$

直观上, $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 中的每个状态变迁都由一个表示事件对的向量标记。$CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 中字符串的左侧组件体现了 G 的行为, 而右侧组件则体现了 G_{nsp}^{aug} 的可观行为。值得注意的是, $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 中某些节点的右侧组件等于 $\emptyset$ 是因为在 $Obs(G_{nsp}^{aug})$ 中存在未定义的变迁。

例 3.2. 再次考虑图 3.1 中所示的自动机 G, 其中 $Q_p^s = \{(1,5),(2,1),(2,6)\}$ 且 $Q_p^{ns} = \{(1,1),(2,2),(2,3),(2,5),(2,7)\}$。并发组合 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 如图 3.3 所示, 其中 G^{aug} 和 $Obs(G_{nsp}^{aug})$ 分别显示在图 3.1 和图 3.2 中。注意, 状态 $(2,3)$ 和 $(2,5)$ 在 G_{nsp}^{aug} 中不存在, 这是由于它们的不可达性。简单起见, 每个 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 中的状态, 其第二个元素按图 3.2 所示的简写表示。

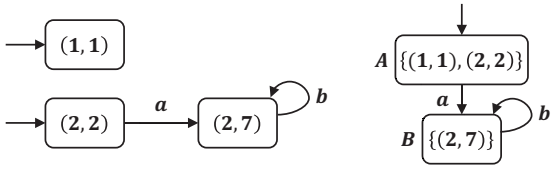

图 **3.2** G^{aug} 的非秘密对子自动机 G_{nsp}^{aug} **(左) ;** 观测器 $Obs(G_{nsp}^{aug})$ **(右)**

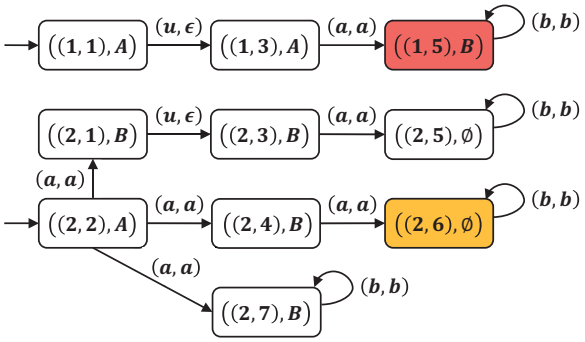

图 **3.3** G^{aug} 和 $Obs(G_{nsp}^{aug})$ 的并发组合 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$

定理 3.1. 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$, 一个投影 P, 一组秘密状态对 Q_p^s, 以及一组非秘密状态对 Q_p^{ns}, G 对于 Q_p^s、Q_p^{ns} 和 P 是PSIFO, 当且仅当对于 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 中的任何状态 $((q_i,q_f), x_{nsp}^{aug})$, 如果 $(q_i,q_f) \in Q_p^s$, 则 $x_{nsp}^{aug} \neq \emptyset$。

证明. "⇐": 假设对于 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 中的任何状态 $((q_i,q_f), x_{nsp}^{aug})$, 若 $(q_i,q_f) \in Q_p^s$, 则 $x_{nsp}^{aug} \neq \emptyset$。对于任意状态 $((q_i,q_f), x_{nsp}^{aug}) \in Q_{cc}$, 如果 $(q_i,q_f) \in Q_p^s$ 且 $x_{nsp}^{aug} \neq \emptyset$, 则自动机 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 中一定存在一条路径

$$((q_i,q_i), x_{nsp}^{0,aug}) \xrightarrow{(e_1,e_1')} ((q_i,q_1), x_{nsp}^{1,aug}) \xrightarrow{(e_2,e_2')} \cdots \xrightarrow{(e_n,e_n')} ((q_i,q_f), x_{nsp}^{aug}).$$

根据定义3.5, 路径中的事件对满足 $e_j = e_j' \in \Sigma_o$ 或 $(e_j,e_j') = (u,\epsilon)$, $j = 1,2,\ldots,n$, 且 $x_{nsp}^{0,aug} \neq \emptyset, x_{nsp}^{1,aug} \neq \emptyset, \ldots, x_{nsp}^{aug} \neq \emptyset$。因此, 根据 G_{nsp}^{aug} 和 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 的构造定义 (定义 3.4 和 3.5) , 存在一条路径 $(q_i',q_i') \xrightarrow{e_1'} (q_i',q_1') \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} (q_i',q_f')$ 在 G^{aug} (或 G_{nsp}^{aug}) 中, 其中 $(q_i',q_i') \in x_{nsp}^{0,aug}, (q_i',q_1') \in x_{nsp}^{1,aug}, \ldots, (q_i',q_f') \in x_{nsp}^{aug}$, 且

$(q_i', q_i'), (q_i', q_1'), \ldots, (q_i', q_f') \in Q_p^{ns}$。因此, 根据 G^{aug} 的定义, G 中存在一条NSPP $q_i' \xrightarrow{e_1'} q_1' \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} q_f'$。另外, 从 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 的构造可以得到一条路径 $q_i \xrightarrow{t} q_f$ 其中 $t = e_1 e_2 \cdots e_n$ 并且 $P(t) = P(e_1' e_2' \cdots e_n')$。因此, G 相对于 Q_p^s、Q_p^{ns} 和 P是PSIFO。

"⇒": 假设 G 相对于 Q_p^s、Q_p^{ns} 和 P 是PSIFO。根据定义 3.2 和 3.3 关于NSPP和PSIFO 的定义, 对于所有满足 $q_i \xrightarrow{t} q_f$ 的 $(q_i, q_f) \in Q_p^s$ 和所有 $t \in L(G, q_i)$, 在 G 中存在一条NSPP $q_i' \xrightarrow{e_1'} q_1' \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} q_f'$, 使得 $P(t') = P(t)$, 其中 $(q_i', q_i'), (q_i', q_1'), \ldots, (q_i', q_f') \in Q_p^{ns}$ 且 $t' = e_1' e_2' \cdots e_n'$。根据定义 3.4, 在 G_{nsp}^{aug} 中存在相应的 NSPP $(q_i', q_i') \xrightarrow{e_1'} (q_i', q_1') \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} (q_i', q_f')$, 因为 $(q_i', q_i'), (q_i', q_1'), \ldots, (q_i', q_f') \in Q_p^{ns}$。根据定义 3.5, 在 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$ 中必然存在一条路径

$$((q_i, q_i), x_{nsp}^{0,aug}) \xrightarrow{(e_1, e_1')} ((q_i, q_1), x_{nsp}^{1,aug}) \xrightarrow{(e_2, e_2')} \cdots \xrightarrow{(e_n, e_n')} ((q_i, q_f), x_{nsp}^{aug}),$$

其中 $e_j = e_j' \in \Sigma_o$ 或 $(e_j, e_j') = (u, \epsilon)$, $j = 1, 2, \ldots, n$, $q_i, q_1, \ldots, q_f \in Q$ 且 $(q_i', q_i') \in x_{nsp}^{0,aug}, (q_i', q_1') \in x_{nsp}^{1,aug}, \ldots, (q_i', q_f') \in x_{nsp}^{aug}$。因此, $x_{nsp}^{0,aug} \neq \emptyset, x_{nsp}^{1,aug} \neq \emptyset, \ldots, x_{nsp}^{aug} \neq \emptyset$ 成立。因此, 对于任何状态 $((q_i, q_f), x_{nsp}^{aug})$, 若 $(q_i, q_f) \in Q_p^s$, 则 $x_{nsp}^{aug} \neq \emptyset$。 □

例 3.3. 考虑图 3.3 中的并发组合 $CC(G^{aug}, Obs(G_{nsp}^{aug}))$。根据定理 3.1, 存在一个状态 $((2, 6), \emptyset)$, 在此状态下秘密被暴露。因此, G 相对于 Q_p^s、Q_p^{ns} 和 P 不是PSIFO。

如下是另一种并发组合的构建, 以简化当 Q_p^{ns} 可以表示为笛卡尔积时PSIFO的验证过程。

定义 3.6. 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$ 和一个可以表示为笛卡尔积的非秘密状态对集合 Q_p^{ns}, 则 G 的非秘密终态子自动机表示为

$$G_{nsf} = (Q_{nsf}, \Sigma, \delta_{nsf}, Q_{nsf,i}),$$

由 G 推导: 通过保留 Q_f^{ns} 中的状态 (Q_p^{ns} 中的右侧组件的集合) , 并随后应用 $Ac(\cdot)$。

定义 3.7. 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$ 和一个可以表示为笛卡尔积的非秘密状态对集合 Q_p^{ns}, $G^{aug} = (Q^{aug}, \Sigma, \delta^{aug}, Q_i^{aug})$ 和 $Obs(G_{nsf}) = (X_{nsf}, \Sigma_o, \delta_{nsf}, x_{nsf,i})$ 的并发组合定义为一个NFA

$$\begin{aligned} CC(G^{aug}, Obs(G_{nsf})) &= (Q_c, \Sigma_c, \delta_c, Q_{c,i}) \\ &= Ac(Q^{aug} \times X_{nsf}, \Sigma_c, \delta_c, Q_i^{aug} \times \{x_{nsf,i}\}), \end{aligned}$$

其中

- $Q_c \subseteq Q^{aug} \times X_{nsf}$ 是状态集合, 其中的每个状态由一个序偶表示, 第一个元素是 G 中的一对状态, 第二个元素是 G_{nsf} 中的一部分状态;
- $\Sigma_c = \{(e,e) : e \in \Sigma_o\} \cup \{(e,\epsilon) : e \in \Sigma_{uo}\}$ 是事件对的集合;
- $Q_{c,i} = Q_i^{aug} \times \{x_{nsf,i}\}$ 是初始状态集;
- $\delta_c : Q_c \times \Sigma_c \to 2^{Q_c}$ 是变迁函数, 对于任意状态 $((q_i,q_f),x_{nsf}) \in Q_c$ 和任意事件 $e \in \Sigma$, 定义如下:
  - 如果 $e \in \Sigma_o$, 则
    - 当 $x_{nsf} \neq \emptyset$ 时,

$$\delta_c(((q_i,q_f),x_{nsf}),(e,e)) = \{((q_i,q'_f),x'_{nsf}) : q'_f \in \delta(q_f,e),$$
$$x'_{nsf} = \delta_{nsf}(x_{nsf},e), \text{ 如果 } \delta_{nsf}(x_{nsf},e) \text{ 已定义 , 否则 } x'_{nsf} = \emptyset\};$$

    - 当 $x_{nsf} = \emptyset$ 时, $\delta_c(((q_i,q_f),\emptyset),(e,e)) = \{((q_i,q'_f),\emptyset) : q'_f \in \delta(q_f,e)\}$;
  - 如果 $e \in \Sigma_{uo}$, 则

$$\delta_c(((q_i,q_f),x_{nsf}),(e,\epsilon)) = \{((q_i,q'_f),x_{nsf}) : q'_f \in \delta(q_f,e)\}.$$

例 3.4. 再次考虑图 3.1 中的自动机 G, 在此示例中, 令 $Q_p^s = \{(1,5),(2,6)\}$ 和 $Q_p^{ns} = \{(2,2),(2,1),(2,3),(2,5),(2,7)\}$。因此, 有 $Q_f^{ns} = \{1,2,3,5,7\}$, 从而得到图 3.4 中的非秘密终态子自动机 G_{nsf}。并发组合 $CC(G^{aug}, Obs(G_{nsf}))$ 如图 3.5 所示, 其中 $Obs(G_{nsf})$ 见图 3.4。

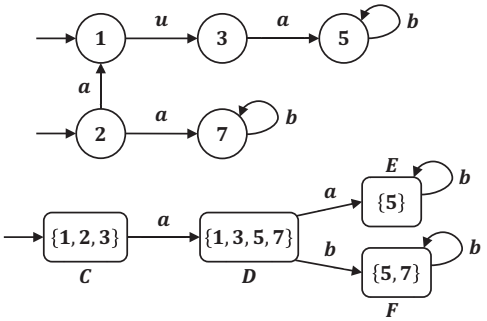

图 **3.4** G 的非秘密终态子自动机 G_{nsf} **(上) ;** 观测器 $Obs(G_{nsf})$ **(下)**

备注 3.1. 注意定义 3.7 中的并发组合概念实际上是定义 3.5 的简化版本, 即当Q_p^{ns}可

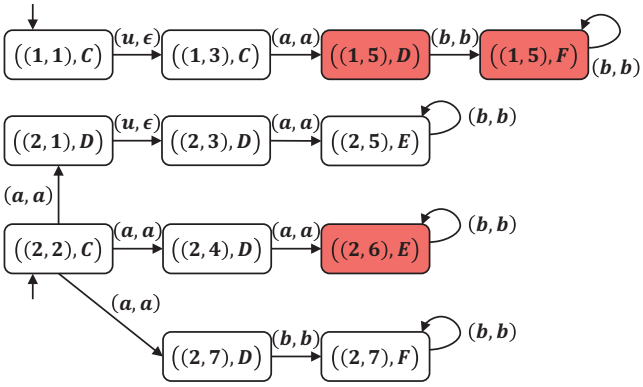

图 **3.5** G^{aug} 和观测器 $Obs(G_{nsf})$ 的并发组合 $CC(G^{aug}, Obs(G_{nsf}))$

表示为笛卡尔积时的特例。并发组合的概念最初由 [122] 提出, 用于非确定性有限自动机 (NFA) 的可检测性验证, 随后被扩展应用于可诊断性验证 [123] 及不透明性验证 [32]。关于该技术在其他领域的应用及最新进展, 可参阅 [124-125] 获取更多细节。此外, 定义 3.5 中的并发组合是 [32] 所提出结构的变体, 二者主要区别在于: 定义 3.5 将并发组合各分量的所有状态均显式表示为状态对。这种表达形式的必要性源于本文考虑的 IFO 需要同时考虑初始状态估计与当前状态估计, 这一特性是 [32] 中并发组合结构所未能涵盖的。

定理 3.2. 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$、一个投影 P、一组秘密状态对 Q_p^s, 以及一组可以表示为笛卡尔积的非秘密状态对 Q_p^{ns}, G 关于 Q_p^s、Q_p^{ns} 和 P 是PSIFO, 当且仅当对于 $CC(G^{aug}, Obs(G_{nsf}))$ 的任何状态 $((q_i, q_f), x_{nsf})$, 如果 $(q_i, q_f) \in Q_p^s$, 则 $x_{nsf} \neq \emptyset$。

证明. "⇐": 假设对于 $CC(G^{aug}, Obs(G_{nsf}))$ 中的任何状态 $((q_i, q_f), x_{nsf})$, 若 $(q_i, q_f) \in Q_p^s$, 则 $x_{nsf} \neq \emptyset$。对于任何状态 $((q_i, q_f), x_{nsf})$, 如果 $(q_i, q_f) \in Q_p^s$ 且 $x_{nsf} \neq \emptyset$, 则自动机 $CC(G^{aug}, Obs(G_{nsf}))$ 中存在一条路径

$$((q_i, q_i), x_{nsf}^0) \xrightarrow{(e_1, e_1')} ((q_i, q_1), x_{nsf}^1) \xrightarrow{(e_2, e_2')} \cdots \xrightarrow{(e_n, e_n')} ((q_i, q_f), x_{nsf}),$$

其中 $e_j = e_j' \in \Sigma_o$ 或 $(e_j, e_j') = (u, \epsilon)$, $j = 1, 2, \ldots, n$, 并且根据定义 3.7, $x_{nsf}^0 \neq \emptyset$, $x_{nsf}^1 \neq \emptyset$, $\ldots$, $x_{nsf} \neq \emptyset$。根据定义 3.6 和 3.7, 在 G (或 G_{nsf}) 中必然存在一条路径 $q_i' \xrightarrow{e_1'} q_1' \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} q_f'$, 其中 $q_i' \in x_{nsf}^0$, $q_1' \in x_{nsf}^1$, $\ldots$, $q_f' \in x_{nsf}$。由于 Q_p^{ns} 可以表示为笛卡尔积, $Q_p^{ns} = Q_i^{ns} \times Q_f^{ns}$ 成立。此外, 由于 $x_{nsf}^0, x_{nsf}^1, \ldots, x_{nsf}$ 是 Q_f^{ns} 的子集, $Q_i^{ns} \times x_{nsf}^0, Q_i^{ns} \times x_{nsf}^1, \ldots, Q_i^{ns} \times x_{nsf} \subseteq Q_p^{ns}$ 成立。注意到 $x_{nsf}^0 = UR(Q_f^{ns} \cap Q_i)$, 因此 $x_{nsf}^0 \cap Q_i^{ns} \neq \emptyset$。设 $q_i' \in x_{nsf}^0 \cap Q_i^{ns}$, 则 (q_i', q_i'), (q_i', q_1'), $\ldots$, $(q_i', q_f') \in Q_p^{ns}$ 成立。

因此, 路径 $q_i' \xrightarrow{e_1'} q_1' \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} q_f'$ 是一条NSPP。此外, 在 $CC(G^{aug}, Obs(G_{nsf}))$ 中存在一条路径 $q_i \xrightarrow{t} q_f$, 其中 $t = e_1e_2\cdots e_n$, 使得 $P(t) = P(e_1'e_2'\cdots e_n')$。因此, G 关于 Q_p^s、Q_p^{ns} 和 P 是PSIFO。

"⇒": 假设 G 关于 Q_p^s、Q_p^{ns} 和 P 是PSIFO。根据定义 3.2 和 3.3, 对于所有 $(q_i, q_f) \in Q_p^s$ 和所有 $t \in L(G, q_i)$, 如果 $q_i \xrightarrow{t} q_f$, 则在 G 中存在一条NSPP $q_i' \xrightarrow{e_1'} q_1' \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} q_f'$, 使得 $P(t') = P(t)$, 其中 (q_i', q_i'), (q_i', q_1'), $\ldots$, $(q_i', q_f') \in Q_p^{ns}$ 且 $t' = e_1'e_2'\cdots e_n'$。根据定义 3.6, 在 G_{nsf} 中存在相应的NSPP $q_i' \xrightarrow{e_1'} q_1' \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} q_f'$, 因为 $q_i', q_1', \ldots, q_f' \in Q_f^s$。根据定义 3.7, 在 $CC(G^{aug}, Obs(G_{nsf}))$ 中存在一条路径

$$((q_i, q_i), x_{nsf}^0) \xrightarrow{(e_1, e_1')} ((q_i, q_1), x_{nsf}^1) \xrightarrow{(e_2, e_2')} \cdots \xrightarrow{(e_n, e_n')} ((q_i, q_f), x_{nsf}),$$

其中 $e_j = e_j' \in \Sigma_o$ 或 $(e_j, e_j') = (u, \epsilon)$, $j = 1, 2, \ldots, n$, $q_i, q_1, \ldots, q_f \in Q$ 且 $q_i' \in x_{nsf}^0$, $q_1' \in x_{nsf}^1$, $\ldots$, $q_f' \in x_{nsf}$。因此, $x_{nsf}^0 \neq \emptyset$, $x_{nsf}^1 \neq \emptyset$, $\ldots$, $x_{nsf} \neq \emptyset$ 成立。因此, 对于任何状态 $((q_i, q_f), x_{nsf})$, 如果 $(q_i, q_f) \in Q_p^s$, 则$x_{nsf} \neq \emptyset$。 □

例 3.5. 考虑图 3.5 中的并发组合 $CC(G^{aug}, Obs(G_{nsf}))$。注意到不存在暴露秘密的状态。因此, 根据定理 3.2, G 是PSIFO。

注意到定理 3.1 和 3.2 不能用于验证定义 3.1 中的IFO, 因为 G_{nsf}^{aug} 和 G_{nsf} 仅保留了并发组合右侧组件中的非秘密状态对的状态。也就是说, 一些保护某些秘密对但包含其他秘密对的路径可能会在 G_{nsf} 和 G_{nsp}^{aug} 中被删除。然而, 本节不仅提出了一个更安全的概念, 还提出了一种比标准 IFO 更高效的验证方法。具体的复杂度分析请参见以下备注。

备注 3.2. 根据定义 3.6–3.7, 构造 $CC(G^{aug}, Obs(G_{nsf}))$ 的复杂度为 $\mathcal{O}(|\Sigma||Q|^3 2^{|Q_f^{ns}|})$, 因为它在最坏情况下有 $|Q|^2 2^{|Q_f^{ns}|}$ 个状态和 $|\Sigma||Q| \times |Q|^2 2^{|Q_f^{ns}|}$ 个变迁。更准确地说, $CC(G^{aug}, Obs(G_{nsf}))$ 中每个状态的变迁数量取决于其左侧组件 G^{aug} 中的变迁数量, 这与 G 中的变迁数量相同。因此, 当 Q_p^{ns} 可以表示为笛卡尔积时, 通过定理 3.2 验证 PSIFO 的复杂度与 $CC(G^{aug}, Obs(G_{nsf}))$ 的大小呈线性关系, 即 $\mathcal{O}(|\Sigma||Q|^3 2^{|Q_f^{ns}|})$。类似地, 通过定理 3.1 验证一般的 Q_p^{ns} 的 PSIFO 的复杂度为 $\mathcal{O}(|\Sigma||Q|^3 2^{|Q_p^{ns}|})$。值得注意的是, $|Q_f^{ns}| \leq |Q|$ 和 $|Q_p^{ns}| \leq |Q|^2$。注意, 当 Q_p^{ns} 可以表示为笛卡尔积时, PSIFO 的验证可以在更快的指数时间内完成。此外, 当 Q_p^{ns} 可以表示为笛卡尔积 (或一般集合) 时, IFO 的验证复杂度为 $\mathcal{O}(|\Sigma|2^{2|Q|})$ (或 $\mathcal{O}(|\Sigma|2^{|Q|^2})$)。因此, 相比于IFO, PSIFO 是一个更安全的概念, 并且具有更高效的验证方法。这

些改进有助于提供更强大、更可靠的安全分析方法。

3.1.3　从强始 (终) 态不透明性到非完全强始-终状态不透明性的归约

本小节首先回顾文献 [32] 中提出的强终态不透明性 (Strong Current-State Opacity, SCSO) 和强初态不透明性 (Strong Initial-State Opacity, SISO) 的概念, 然后讨论它们与PSIFO的关系。

一个 NFA $G = (Q, \Sigma, \delta, Q_i)$ 被认为是关于一组秘密状态 $Q_s \subseteq Q$、一组非秘密状态 $Q_{ns} \subseteq Q$ 和一个投影 P 强终态不透明的 (或强初态不透明的) , 如果对于所有 $q_i \in Q_i$ (或 $q_i \in Q_i \cap Q_s$) 和所有 $t \in L(G, q_i)$, 当 $\delta(q_i, t) \cap Q_s \neq \emptyset$ (或 $\delta(q_i, t) \neq \emptyset$) 时, 存在一条非秘密路径[①] $q_i' \xrightarrow{t'} q$ 使得 $P(t') = P(t)$, 其中 $q_i' \in Q_i$ (或 $q_i' \in Q_i \cap Q_{ns}$) 且 $q \in Q$。

不难得出结论, SCSO 和 SISO 都是 PSIFO 的特例。更具体地说, 类似于文献 [18] 中的结果, 通过设置 $Q_p^s = Q_i \times Q_s$ 和 $Q_p^{ns} = Q_i \times Q_{ns}$ 可以将 SCSO 问题转化为 PSIFO 问题。此外, 通过设置 $Q_p^s = Q_s \times Q$ 和 $Q_p^{ns} = Q_{ns} \times Q$ 将 SISO 问题转化为 PSIFO 问题。换句话说, SCSO 可以看作是 PSIFO 的一个特例, 其中初态对结果没有影响。类似地, SISO 可以看作是 PSIFO 的另一个特例, 其中终态不发挥作用。然而, 值得强调的是, 接下来将介绍的 SIFO 概念与 SCSO 和 SISO 的概念不可比。这意味着 SIFO 的验证是一个更具挑战性的问题。

3.2　强始-终状态不透明性

本节定义了一个安全要求高于 PSIFO 的新概念, 即 NFA 的**强始-终态不透明性** (Strong Initial-and-Final-State Opacity, SIFO) 。此外, 本节还提出了相应的验证方法以及 SIFO 与 PSIFO 之间的关系。

3.2.1　强始-终状态不透明性的概念

首先引入强非秘密对路径 (Strongly Non-Secret-Pair Path, SNSPP) 的概念。

定义 3.8 (SNSPP)**.** 在 NFA G 中, 路径 $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$ 被称为 SNSPP, 如果它满足以下条件: 对于所有 $j = 0, 1, \ldots, n$ 和 $k = 0, 1, \ldots, n$, 如果 $q_j \in Q_i$, 则 $(q_j, q_k) \in Q_p^{ns}$ 且 $j \leq k$。

① 在 NFA G 中, 一条路径 $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$ 被称为非秘密路径, 如果 $q_j \in Q_{ns}$ 对于 $j = 0, 1, \ldots, n$。

定义 3.9 (SIFO)**.** 给定一个 NFA $G = (Q, \Sigma, \delta, Q_i)$、一个投影 P、一组秘密状态对 Q_p^s 和一组非秘密状态对 Q_p^{ns}, G 被认为相对于 Q_p^s、Q_p^{ns} 和 P 是强始-终状态不透明的 (Strongly Initial-and-Final-State Opaque, SIFO) , 如果对于所有 $(q_i, q_f) \in Q_p^s$ 和所有 $t \in L(G, q_i)$, 满足 $q_f \in \delta(q_i, t)$ 的情况下, 存在一条SNSPP $q_i' \xrightarrow{t'} q_f'$ 使得 $P(t') = P(t)$。

例 3.6. 仍然考虑图 3.1 中描绘的自动机 G, 在这个例子中, 设 $Q_p^s = \{(1,3),(2,6)\}$ 和 $Q_p^{ns} = \{(2,2),(2,1),(2,3),(2,5)\}$。根据定义 3.1 和 3.3, G 在 Q_p^s、Q_p^{ns} 和 P 的作用下是 (PS)IFO。然而, 它并不是SIFO, 因为路径 $2 \xrightarrow{a} 1 \xrightarrow{u} 3 \xrightarrow{a} 5$ 可能会无意中暴露秘密状态对 $(1,3)$。请注意, 状态 1 是这条路径经过的初始状态之一, 但不是这条路径的起点。

假设存在一个多智能体系统, 建模为NFA, 其中代理 2 通过与代理 5 的通信成功地保护了它与代理 6 通信的秘密。然而, 它不小心暴露了代理 1 与代理 3 通信的秘密。因此, 这提出了比 PSIFO 更高的安全要求, 引申出了定义 3.9 。从定义 3.1、3.3 和 3.9 可以很容易地推导出, 如果 G 是SIFO, 那么它也是 (PS)IFO。然而, 反之则不成立。

3.2.2 强始-终状态不透明性的验证

与之前的情况类似, 验证过程被分为两种情况: 当 Q_p^{ns} 是一般集合时, 以及当 Q_p^{ns} 可以表示为笛卡尔积时。之后将说明更高的安全要求意味着更高的验证复杂度。众所周知, 在有向有环图中, 找到两个给定节点之间的所有路径在实际操作中几乎是不可能的。然而, 使用深度优先搜索 (Depth First Search, DFS) 算法[126] 可以以指数复杂度找到所有简单路径①。以下命题简化了搜索过程, 并得出了定义 3.10。这两个命题的证明非常直观, 因此省略。为了方便, 这里将SNSPP的概念从 G 扩展到 G^{aug}。

命题 3.1. 如果在 G 中存在一条SNSPP $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$, 当且仅当在其扩展自动机 G^{aug} 中存在一条对应的SNSPP $(q_0, q_0) \xrightarrow{e_1} (q_0, q_1) \xrightarrow{e_2} \cdots \xrightarrow{e_n} (q_0, q_n)$。

命题 3.2. 检验两个给定状态之间是否存在一条SNSPP可以简化为检验这两个状态之间是否存在一条简单路径, 这条简单路径是SNSPP。

① 简单路径是指不经过重复节点的路径。

定义 3.10. 给定一个 NFA $G = (Q, \Sigma, \delta, Q_i)$ 和一组非秘密状态对 Q_p^{ns}, 改良的非秘密对子自动机 $G_{nsp}^{aug\prime} = (Q_{nsp}^{aug\prime}, \Sigma, \delta_{nsp}^{aug\prime}, Q_{nsp,i}^{aug})$ 通过从 G_{nsp}^{aug} 中移除所有不能通过简单的SNSPP到达的状态及其相关的变迁得到。

仅保留 G_{nsp}^{aug} 中通过至少一条SNSPP可达的状态, 从而得到 $G_{nsp}^{aug\prime}$。在后者中, 按照定义 3.5, 构造并发组合 $CC(G^{aug}, Obs(G_{nsp}^{aug\prime}))$。

备注 3.3. 定义 3.10 中的描述被简化了, 因为具体过程可以很容易地设想出来。现在, 本注释明确描述定义 3.10 中操作的计算复杂度。构建过程中, 在 G_{nsp}^{aug} 中的两个给定状态之间的所有简单路径都执行一次DFS。DFS 遍历可能会多次回溯到某个节点, 因为它在探索图的不同分支。因此, 复杂度为:

$$\mathcal{O}\big((|Q_p^{ns}| + |Q_p^{ns}| \times |Q||\Sigma|) \times \mathcal{P}\big), \tag{3-1}$$

其中 $\mathcal{P}$ 是两个给定节点之间的简单路径数量。G_{nsp}^{aug} 中的变迁数量是 $|Q_p^{ns}| \times |Q||\Sigma|$, 而不是 $|Q_p^{ns}|^2|\Sigma|$, 因为 G_{nsp}^{aug} (或 G^{aug}) 中从一个状态出发的变迁数量与从 G 中一个状态出发的变迁数量在同一数量级。在最坏情况下, $\mathcal{P}$ 由以下公式给出:

$$\sum_{k=1}^{|Q|-1} \begin{pmatrix} |Q_p^{ns}| - 2 \\ k - 1 \end{pmatrix},$$

其中 k 是路径的长度。更准确地说, 这是通过从余下 $|Q_p^{ns}| - 2$ 个状态中选择 $k-1$ 个不同的状态得到的。路径的最大长度为 $|Q| - 1$, 因为 G^{aug} 的进程与 G 的进程相同。在获得所有简单路径后, 检查这些简单路径中是否存在一条SNSPP, 其复杂度为:

$$\mathcal{O}\big(|Q_i||Q_p^{ns}| \times \mathcal{P} \times (k+1)^2\big). \tag{3-2}$$

最终, 构造定义 3.10 中的 $G_{nsp}^{aug\prime}$ 的复杂度表示为 $|Q_i||Q|$ 乘以公式 (3-1) 和公式 (3-2) 的和, 因为存在 $|Q_i||Q|$ 种源节点和目标节点的组合。

定理 3.3. 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$、一个投影 P、一组秘密状态对 Q_p^s 和一组非秘密状态对 Q_p^{ns}, G 相对于 Q_p^s、Q_p^{ns} 和 P 是SIFO, 当且仅当对于 $CC(G^{aug}, Obs(G_{nsp}^{aug\prime}))$ 的任何状态 $((q_i, q_f), x_{nsp}^{aug\prime})$, 如果 $(q_i, q_f) \in Q_p^s$, 则 $x_{nsp}^{aug\prime} \neq \emptyset$。

证明. 由于这个证明与定理 3.1 的证明类似, 可以通过结合定义 3.5、3.8、3.9、3.10 和性质 3.1、3.2 来轻松推导, 因此省略。 □

例 3.7. 再次考虑图 3.1 中描绘的自动机 G, 在此示例中, 设 $Q_p^s = \{(2,6)\}$ 和 $Q_p^{ns} = \{(2,2),(2,1),(2,3),(2,5),(1,1),(1,3),(1,5)\}$。并且, 图 3.7 中展示了并发组合 $CC(G^{aug}, Obs(G_{nsp}^{aug\prime}))$, 其中 $Obs(G_{nsp}^{aug\prime})$ 如图 3.6 所示。请注意, 此例中$G_{nsp}^{aug\prime}$ 和 G_{nsp}^{aug} 的结构是相同的。状态 $(2,1),(2,3),(2,5)$ 的存在是由于 Q_p^{ns} 中存在 $(1,1),(1,3),(1,5)$。状态 $\big((2,6),J\big)$ 表示状态 $\big((2,6),\{(2,5)\}\big)$, 它成功地保护了秘密。因此, 根据定理 3.3, G 是SIFO。

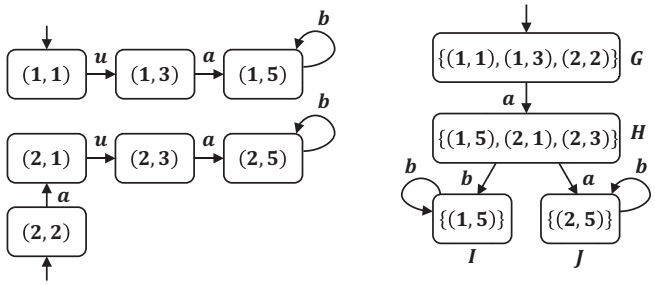

图 **3.6** 改良的非秘密对子自动机 $G_{nsp}^{aug\prime}$ **(左) ;** 观测器 $Obs(G_{nsp}^{aug\prime})$ **(右)**

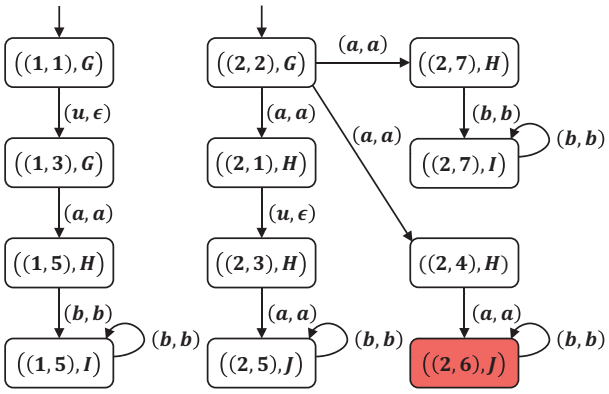

图 **3.7** G^{aug} 与观测器 $Obs(G_{nsp}^{aug\prime})$ 的并发组合 $CC(G^{aug}, Obs(G_{nsp}^{aug\prime}))$

现在考虑当 Q_p^{ns} 可以表示为笛卡尔积时的 SIFO 验证。

定义 3.11. 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$ 和一组可以表示为笛卡尔积的非秘密状态对 Q_p^{ns}, 改良的非秘密终态子自动机 $G'_{nsf} = (Q'_{nsf}, \Sigma, \delta'_{nsf}, Q_{nsf,i})$ 是通过从 G_{nsf} 移除所有无法通过简单的SNSPP到达的状态及其相关的变迁而得到的。

类似于前述方法, 仅保留在 G_{nsf} 中通过至少一条SNSPP可达的状态, 从而得到 G'_{nsf}。在 G'_{nsf} 中, 用与定义 3.7 相同的方式构造 $CC(G^{aug}, Obs(G'_{nsf}))$。

定理 3.4. 给定一个NFA $G = (Q, \Sigma, \delta, Q_i)$、一个投影 P、一组秘密状态对 Q_p^s 和一组可以表示为笛卡尔积的非秘密状态对 Q_p^{ns}, G 相对于 Q_p^s、Q_p^{ns} 和 P 是SIFO, 当

且仅当对于 $CC(G^{aug}, Obs(G'_{nsf}))$ 的任何状态 $((q_i, q_f), x'_{nsf})$, 如果 $(q_i, q_f) \in Q_p^s$, 则 $x'_{nsf} \neq \emptyset$。

证明. 该证明被省略, 因为它与定理 3.2 的证明类似, 并且可以通过结合定义 3.8、3.9、3.7、3.11 和性质 3.2 来轻松推导。 □

例 3.8. 再次考虑图 3.1 中描绘的自动机 G, 其中 $Q_p^s = \{(1,5),(2,6)\}$ 和 $Q_p^{ns} = \{(2,2),(2,1),(2,3),(2,5),(2,7)\}$ 与例 3.4 中的相同。并发组合 $CC(G^{aug}, Obs(G'_{nsf}))$ 如图 3.9 所示, 其中 $Obs(G'_{nsf})$ 显示在图 3.8 中。从状态 2 到状态 1 没有SNSPP, 因此状态 1 不在 G'_{nsf} 中。根据定理 3.4, G 不是SIFO。

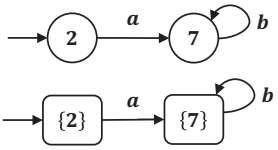

图 **3.8** G_{nsf} 的改良非秘密终态子自动机 G'_{nsf} (上)**;** 观测器 $Obs(G'_{nsf})$ (下)

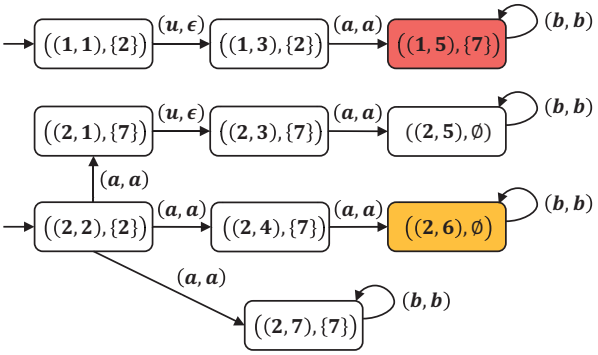

图 **3.9** G^{aug} 与观测器 $Obs(G'_{nsf})$ 的并发组合 $CC(G^{aug}, Obs(G'_{nsf}))$

需要注意的是, 构造 $CC(G^{aug}, Obs(G'_{nsf}))$ 的计算复杂度在最坏情况下等同于构造 $CC(G^{aug}, Obs(G^{aug\prime}_{nsp}))$ 的复杂度。这是因为 G'_{nsf} 的构造复杂度与 $G^{aug\prime}_{nsp}$ 的构造复杂度相同。

3.2.3　一个更高效的强始-终状态不透明性验证算法

以下命题探讨了在特定情况下 PSIFO 和 SIFO 之间的等价性。

命题 3.3. 给定一个投影 P、一组秘密状态对 Q_p^s 和一组可以表示为笛卡尔积的非秘密状态对 Q_p^{ns}, 如果条件 $q \in Q_i \Rightarrow q \in Q_i^{ns}$ 成立, 则 NFA G 相对于 Q_p^s、Q_p^{ns} 和 P 是PSIFO, 当且仅当它相对于 Q_p^s、Q_p^{ns} 和 P 是SIFO。

证明. “⇒”: 假设 G 相对于 Q_p^s、Q_p^{ns} 和 P 是PSIFO。根据定义 3.2 和 3.3, 对于所有 $(q_i, q_f) \in Q_p^s$ 和所有 $t \in L(G, q_i)$, 其中 $q_i \xrightarrow{t} q_f$, 存在一条NSPP $q_i' \xrightarrow{e_1'} q_1' \xrightarrow{e_2'} \cdots \xrightarrow{e_n'} q_f'$, 使得 $P(t') = P(t)$, 其中 $q_i', q_1', \ldots, q_f' \in Q_f^{ns}$, 且 $t' = e_1'e_2' \cdots e_n'$。由于 Q_p^{ns} 可以表示为笛卡尔积, $Q_i^{ns} \times \{q_i', q_1', \ldots, q_f'\} \subseteq Q_p^{ns}$ 成立。因此, 根据 $q \in Q_i \Rightarrow q \in Q_i^{ns}$, $\{q_j'\} \times \{q_i', q_1', \ldots, q_f'\} \subseteq Q_p^{ns}$ 成立, 其中 $q_j' \in Q_i = Q_i^{ns}$。进一步地, $\{q_j'\} \times \{q_j', q_{j+1}', \ldots, q_f'\} \subseteq Q_p^{ns}$ 成立。设 $q_k' = q_j', q_{j+1}', \ldots, q_f'$, $(q_j', q_k') \in Q_p^{ns}$ 成立, 其中 $q_j' \in Q_i$。结合定义 3.8 和 3.9, NSPP $q_i' \xrightarrow{e_1} q_1' \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_f'$ 也是一条SNSPP, 这意味着 G 是SIFO。

“⇐”: 根据定义 3.3 和 3.9, 只需证明当 Q_p^{ns} 可以表示为笛卡尔积时, 一条SNSPP $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$ 也是一条NSPP。选择 $q_j = q_0$, 根据定义 3.8, 对于 $k = 0, 1, \ldots, n$, $(q_0, q_k) \in Q_p^{ns}$ 都成立。因此, G 是PSIFO。 □

根据性质 3.3, 可以明显看出, 当非秘密对集合 Q_p^{ns} 可以表示为笛卡尔积, 并且同时满足蕴含条件 $q \in Q_i \Rightarrow q \in Q_i^{ns}$ 时, 定理 3.2 相同的陈述也可以用来验证SIFO。

3.3 本章小结

在 DES 中, IFO 是描绘系统安全性的重要性质之一, 本章介绍了其两个增强版本: PSIFO 和 SIFO。这些概念在各种应用场景中, 如网络通信系统和多智能体系统, 提供了更高的保密能力。此外, 本章提出了两种新的并发组合, 这些组合能够验证这两种强始-终状态不透明性。PSIFO 的验证需要指数级复杂度, 但相比于IFO, 相对较低。然而, 对于一般的 NFA, 验证 SIFO 的复杂度甚至更高。因此, 本章提出了一个条件来提高 SIFO 验证的效率。最后, 本章展示了从 SCSO 和 SISO 到 PSIFO 的归约, 证明了 PSIFO 和 SIFO 的重要性和普遍性。下一章将研究 DES 中描绘系统安全性的另一个重要性质: 临界可观测性, 并考虑其控制器综合问题。

第4章　模块化系统的临界可观测性使能

4.1　基于语言的临界可观测性描述

本节研究**临界可观测性** (Critical Observability, CO) 的性质, 并提出其在DFA 框架下的语言特征描述。CO 的标准定义如下。

定义 4.1 (CO[44]). 一个 DFA $G = (Q, \Sigma, \delta, q_0, Q_m)$ 对于投影 $P : \Sigma^* \to \Sigma_o^*$ 和**关键状态集** $Q_c \subseteq Q$ 是临界可观的 (CO), 当且仅当对于每一个 $s \in L(G)$, 都有

$$\delta(q_0, P^{-1}[P(s)]) \subseteq Q_c \vee \delta(q_0, P^{-1}[P(s)]) \subseteq Q \setminus Q_c,$$

其中 $\delta(I, P^{-1}P(w)) = \bigcup_{v \in P^{-1}P(w)}$。

临界可观测性的概念要求在给定系统的任何观测结果时, 都能够明确判断系统当前是否处于某些关键状态。

注意, CO与一种称为**常态性** (Normality) 的基于语言的性质之间存在联系。回忆常态性的公式：$\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$。由于 $\overline{K} \subseteq P^{-1}[P(\overline{K})]$, 假设 $K \subseteq L(G)$, 所以包含关系 $\overline{K} \subseteq P^{-1}[P(\overline{K})] \cap L(G)$ 总是成立。换句话说, 常态性条件可以重写为以下包含关系：

$$P^{-1}[P(\overline{K})] \cap L(G) \subseteq \overline{K}.$$

有时, 常态性也使用等价的基于字符串的表述形式, 在这种表述中, 常态性等价于以下的蕴涵关系 (由上述包含关系给出) :

$$\forall s, s' \in \Sigma^*, s \in L(G) \wedge s' \in \overline{K} \wedge P(s') = P(s) \Rightarrow s \in \overline{K}.$$

另一种等价的常态性表述如下：

$$\forall s, s' \in \Sigma^*, s \in L(G) \setminus \overline{K} \wedge s' \in \overline{K} \Rightarrow P(s') \neq P(s). \tag{4-1}$$

这意味着, 常态性要求对于系统生成的任意两个字符串 (一个是安全的, 另一个是非安全的) , 应能够通过观测区分它们。CO 涉及区分系统中两种行为的能力：关键和非关键。然而, CO 是一种基于状态的性质, 其中某些状态被定义为关键状态集。显然, DFA 的 CO 可以完全通过以下语言性质来刻画, 该性质类似于常态性, 但在定义中没有前缀闭包：

定义 4.2 (语言临界可观测性 (LCO)). 对于语言 $K \subseteq L(G) \subseteq \Sigma^*$, 若

$$K = P^{-1}[P(K)] \cap L(G),$$

则称 K 相对于 $L(G)$ 和 $P : \Sigma^* \to \Sigma_o^*$ 是语言临界可观的 (LCO) 。

更准确地说, CO可以用以下语言刻画。

命题 4.1. 设 $G = (Q, \Sigma, \delta, q_0, Q_m)$ 为一个DFA, $Q_c \subseteq Q$ 为一个关键状态集。设 K 为一组字符串, 定义为 $K = \{s' \in L(G) : \delta(q_0, s') \in Q_c\}$。则集合 K (和 $L(G) \setminus K$) 关于 $L(G)$ 和 P 是LCO 当且仅当 G 关于 $P : \Sigma^* \to \Sigma_o^*$ 和 Q_c 是CO。

证明. 注意到, 对于DFA, CO的定义可以重写为如下形式：

$$\forall s, s' \in L(G), \delta(q_0, s) \in Q \setminus Q_c \wedge \delta(q_0, s') \in Q_c \Rightarrow P(s) \neq P(s'). \tag{4-2}$$

则对于 $K = \{s' \in L(G) : \delta(q_0, s') \in Q_c\}$, 蕴含式 (4-1) 和 (4-2) 是等价的。 □

可以通过检验 $P^{-1}P(K) \cap L$ 是否包含在 L 中来在多项式时间内验证LCO[45]。常态性与LCO的区别在于, 常态性使用了 K 的前缀闭包, 而LCO直接使用 K。因此, 以下推论成立。

推论 4.1. 设 $G = (Q, \Sigma, \delta, q_0, Q_m)$ 为一个DFA, $Q_c \subseteq Q$ 为一个关键状态集, 使得 $K = \{s' \in L(G) : \delta(q_0, s') \in Q_c\}$ 是前缀闭合的。则 K 关于 $L(G)$ 和 $P : \Sigma^* \to \Sigma_o^*$ 是常态的当且仅当 G 关于 P 和 Q_c 是CO。 □

值得注意的是, 命题 4.1 对于 $K = \{s \in L(G) : \delta(q_0, s) \in Q \setminus Q_c\}$ 也成立。然而, 在 NFA 的框架下, 这一命题并不成立。为了说明命题 4.1 不适用于非确定性, 考虑以下示例。

例 4.1. 考虑一个 NFA $G = (Q, \Sigma, \delta, Q_i, Q_m)$, 其中 $Q = \{0, 1, 2\}$, $\Sigma = \Sigma_o = \{a\}$, $Q_i = \{0\}$, 以及两个变迁 $\delta(0, a) = \{1, 2\}$。设 $Q_c = \{0, 1\}$ 为关键状态集。则 G 不是 CO, 因为 $1 \in Q_c$ 但 $2 \in Q \setminus Q_c$。然而, 语言 $K = \{\varepsilon, a\} = \overline{K} = L(G)$ 既是 LCO, 也是常态的。注意, 对于 NFA, CO 蕴含 LCO。

从常态性和LCO的定义中可以明显看出, $K \subseteq L(G)$ 关于 $L(G)$ 和 P: $\Sigma^* \to \Sigma_o^*$ 是常态的, 当且仅当 $\overline{K}$ 关于 $L(G)$ 和 P: $\Sigma^* \to \Sigma_o^*$ 是 LCO。由于常态性是在语言的前缀闭包上定义的, 因此K 是常态的当且仅当 $\overline{K}$ 也是常态的。然而, $\overline{K}$ 是常态的

当且仅当它是 LCO, 因为 LCO 并不是在语言的前缀闭包上定义的。请注意, 常态性并不一定蕴含 LCO, 参见如下示例。

例 4.2. 设 $\Sigma = \{a, \tau\}$, $\Sigma_o = \{a\}$, $L(G) = \overline{\{\tau a, a\tau\}}$, 并且 $K = \{\tau a, a\tau\}$。在这种情况下, 由于 $\overline{K} = L(G)$, 显然 K 是常态的。然而, K 不是 LCO。这是因为考虑 $s = a$ 和 $s' = a\tau$ 时, $P(s) = P(s')$, 有 $s' \in K$ 但 $s \in L(G) \setminus K$。

另一方面, 值得注意的是, LCO也不一定蕴含常态性。只有当 $P^{-1}[P(K)]$ 和 $L(G)$ 是非冲突语言时, LCO 才可能蕴含常态性。然而, 这一条件并不总是适用, 因为一般情况下, 仅 $\overline{P^{-1}[P(K)] \cap L(G)} \subseteq \overline{P^{-1}[P(K)]} \cap \overline{L(G)}$ 成立。反向的包含关系只有在 $P^{-1}[P(K)]$ 和 $L(G)$ 是非冲突语言时才成立。因此, 常态性和 LCO, 即常态性和 CO, 在一般 (非前缀闭合) 的语言下是不可比的。以下示例表明, LCO 并不蕴含常态性。

例 4.3. 设 $\Sigma = \{a, \tau\}$, $\Sigma_o = \{a\}$, $L(G) = \overline{\{\tau, a\tau\}}$, 并且 $K = \{a, a\tau\}$。则 K 是 LCO, 因为 $P^{-1}[P(K)] \cap L(G) = K$。然而, K 不是常态的, 因为存在字符串 $s = \varepsilon \in \overline{K}$ 和 $s' = \tau \in L \setminus \overline{K}$, 且 $P(s) = P(s')$, 这违反了常态性的要求。

很容易看出, 类似于常态性, LCO 在语言的并集下仍保持。因此, 以下引理成立。

引理 4.1. 设 $K_i \subseteq L(G) \subseteq \Sigma^*$ 对于 $i \in I$ 是一个任意的语言族, 且这些语言相对于 $L(G)$ 和 $P : \Sigma^* \to \Sigma_o^*$ 是 LCO。则 $\bigcup_{i\in I} K_i$ 相对于 $L(G)$ 和 P 也是 LCO。

证明. 由于投影和逆投影对于语言并集满足分配律, 因此:

$$P^{-1}[P(\bigcup_{i\in I} K_i)] \cap L(G) = \bigcup_{i\in I} P^{-1}[P(K_i)] \cap L(G).$$

由于对于任意 $i \in I$, $P^{-1}[P(K_i)] \cap L(G) = K_i$ 成立, 因此 $\bigcup_{i\in I} K_i$ 也是 LCO。 □

一个直接的结果是, 对于任何 $K \subseteq L(G)$, 其唯一的极大临界可观子语言, 即所有(L)CO 子语言的并集, 总是存在。然而, 这个子语言不一定是可观的。由于可观测性像常态性一样是在前缀闭包上定义的, LCO 并不蕴含可观测性, 见如下示例。

例 4.4. 设 $\Sigma = \{a, b, \tau\}$, $\Sigma_o = \Sigma_c = \{a, b\}$, $L(G) = \overline{\{ab\tau, \tau a, \tau b\}}$, 且 $K = \{ab, ab\tau, \tau b\}$ $= L(G) \setminus \{\varepsilon, a, \tau, \tau a\}$。在这种情况下, K 仍然是 LCO, 但它不是可观的。这一点很明显, 因为考虑 $s = \varepsilon \in \overline{K}$ 和 $s' = \tau \in \overline{K}$ 时, 有 $sa = a \in \overline{K}$ 和 $s'a = \tau a \in L(G)$, 但 $\tau a \notin \overline{K}$, 并且 $P(s) = P(s') = \varepsilon$。

由于示例 4.2 中的语言是常态的, 因此也是可观的, 故可以得出结论, LCO 和可观测性也是不可比的。

4.2 极大可控、常态和临界可观子语言的计算

接下来的部分将重点关注DFA。根据命题 4.1, 关键 (或非关键) 状态集和**关键语言** K (或**非关键语言** $L \setminus K$) 可以互换。本节中定义的控制操作将生成关键 (非关键) 状态集的 (适当的) 子集, 这些子集将在生成的 DFA 中成为新的关键 (非关键) 状态集, 在这些 DFA 中, LCO 得到满足, 从而使能 CO。

给定一个 DFA $G = (Q, \Sigma, \delta, q_0)$ 及其语言 $L = L(G)$ 和一组关键状态 $Q_c \subseteq Q$。设 $Q_m = Q_c$, 根据命题 4.1, G 关于 P 和 Q_c 是 CO 当且仅当 $K = L_m(G)$ 是 LCO。然而, 命题中还有第二种选项, 即 G 关于 P 和 Q_c 是 CO 当且仅当 $L \setminus K$ 是 LCO。对于 DFA, 采用这两种互补的方法来使能CO是有用的, 因为通常基于关键语言的方法会产生过于严格的解, 而基于非关键语言的方法可能会产生令人满意的解, 见示例 4.6, 反之亦然。由于 DFA 的 LCO 与 CO 等价, 本文其余部分稍微滥用符号, 仅考虑 LCO, 同时将 K 关于 L 和 P 的**极大临界可观子语言** (Supremal Critically-Observable Sublanguage) 记作 $\mathrm{supCO}(K, L, P)$。

为了在 DFA 中使能CO, 需要计算 $\mathrm{supCO}(K, L, P)$ 或 $\mathrm{supCO}(L \setminus K, L, P)$。这两个与 K 最“接近的” LCO 语言, 可以形式化如下。注意到 LCO 在任意交集下保持。即, 如果 $K_i \subseteq L(G) \subseteq \Sigma^*, i \in I$, 是相对于 $L(G)$ 和 $P : \Sigma^* \to \Sigma_o^*$ 的 LCO 语言, 则 $\bigcap_{i \in I} K_i$ 也是相对于 $L(G)$ 和 P 的 LCO 语言。这意味着总是存在 K 的**极小临界可观超语言** (Infimal Critically-Observable Superlanguage) , 记作

$$\mathrm{infCO}(K, L, P) = \inf\{M \supseteq K : K \text{ 是相对于 } L \text{ 和 } P \text{ 的 LCO}\},$$

它等于所有临界可观超语言的交集。故得到如下结果。

命题 4.2. $\mathrm{supCO}(L \setminus K, L, P) = L \setminus \mathrm{infCO}(K, L, P)$.

证明. “$\supseteq$”: 显然, $L \setminus \mathrm{infCO}(K, L, P) \subseteq L \setminus K$。因此还需证明 $L \setminus \mathrm{infCO}(K, L, P)$ 相对于 L 和 P 是 LCO。由于 $\mathrm{infCO}(K, L, P)$ 相对于 L 和 P 是 LCO, 根据命题 4.1, 它的补集也是 LCO。

“$\subseteq$”: 显然, $L \setminus \mathrm{supCO}(L \setminus K, L, P) \supseteq K \supseteq \mathrm{infCO}(K, L, P)$ 成立。因此还需证明 $L \setminus \mathrm{supCO}(L \setminus K, L, P)$ 相对于 L 和 P 是 LCO。由于 $\mathrm{supCO}(L \setminus K, L, P)$ 相对

于 L 和 P 是 LCO, 根据命题 4.1, 它的补集也是 LCO。 □

然而, 这些记作 K 的新的规范不一定可以通过监督控制实现。因此, 需要计算 $\mathrm{supCNCO}(K, L, P)$ 或 $\mathrm{supCNCO}(L \setminus K, L, P)$, 这些可能不一定是前缀闭合的。这里, $\mathrm{supCNCO}(K, L, P)$ 表示相对于 L 和 P 的 K 的极大可控、常态和临界可观子语言。

将 [2] 中原本为计算非前缀闭规范的极大常态和可控子语言设计的算法, 修改后用于计算 $\mathrm{supCNCO}(K, L, P)$, 如算法 4.1 所示。不失一般性地, 假设系统 G 是相对于 P 的状态分区自动机 (State Partition Automata, SPA) ①, 这总是能通过令 $G = G \parallel Obs(G)$[2] 来保证。显然, 算法 4.1 在有限次迭代后停止, 输出子语言的极大性由 [39] 中的引理 2.1 保证。

算法 4.1 计算$\mathrm{supCNCO}(K, L, P)$

输入: 满足 $L = L(G)$ 的 DFA $G = (Q, \Sigma, \delta, q_0, Q_m)$。
满足 $L_m(H) = K$, $L(H) = \overline{K}$ 的 DFA $H = (X, \Sigma, f, x_0, Q_m)$, 其中 $K \subseteq L(G)$.
输出: 子语言 $\mathrm{supCNCO}(K, L, P)$。

1: 构建观测器 $Obs(G) = (Y, \Sigma_o, \xi, y_0)$。
2: 令 $H_0 = H$, 其中 $H_0 = (X_0, \Sigma, f_0, x_0, Q_{0,m})$, 设$i = 0$。
3: 计算以下条件:

- 可控性: $X_i^C = X_i \setminus \{x_i \in X_i \mid \exists s_{uc} \in E_{uc}^* : \delta(x_i, s_{uc}) \notin X_i\}$;
- 常态性: $X_i^N = \{x_i \in X_i \mid x_i \in y \in Y : y \subseteq X_i\}$;
- 临界可观测性: $X_i^{CO} = \{x_i \in X_i \mid x_i \in y \in Y : y \subseteq Q_m \vee y \subseteq Q \setminus Q_m\}$.

4: 计算

$$
\begin{aligned}
X_i' &= X_i^C \cap X_i^N \cap X_i^{CO}, \\
f_i' &= f \mid X_i^C \cap X_i^N \cap X_i^{CO②}, \\
X_{i,m}' &= X_{i,m} \cap X_i^C \cap X_i^N \cap X_i^{CO}.
\end{aligned}
$$

5: 令 $H_{i+1} = Trim(X_i', \Sigma, f_i', x_0, X_{i,m}')$③。
6: **if** $L_m(H_{i+1}) = L(H_{i+1}) = \emptyset$ **then**
7: **return** $\mathrm{supCNCO}(K, L, P) = \emptyset$.
8: **else if** $H_{i+1} = H_i$ **then**
9: **return** $\mathrm{supCNCO}(K, L, P) = L_m(H_i)$.
10: **else**
11: 令 $i = i + 1$。
12: 回到步骤 3。

① 如果自动机 G 的观测器中的任何两个状态要么相同, 要么它们的交集为空, 则 G 被认为是SPA。

例 4.5. 首先通过省略算法 4.1 中步骤 3 的可控性和常态性条件来说明命题 4.2 (即计算 $\mathrm{supCO}(K, L, P)$) 。考虑如图4.1 (左) 所示的 DFA G, 其关键状态集为 $Q_c = \{5\}$。根据命题 4.1, 设 $K = \{s' \in L(G) : \delta(q_0, s') \in Q_c\} = \{cc\}$, 因此 $H_0 = H$ 如图 4.2 (左) 所示。在步骤 1 中, 算法 4.1构建观测器 $Obs(G)$, 如图 4.1 (右) 所示。对于第一轮 ($i = 0$) , 可以得到 $X_0^{CO} = X_0' = \{1, 2\}$ 和 $X_{0,m}' = \{5\} \cap \{1, 2\} = \emptyset$。因此, $\mathrm{supCO}(K, L, P) = \emptyset$。请注意, 根据命题 4.2 将 K 和 $L \setminus K$ 互换, 可以得到了对偶解, 即 $\mathrm{infCO}(L \setminus K, L, P) = L \setminus \mathrm{supCO}(K, L, P) = L$ 是相应的非关键语言。

根据命题 4.1, $\mathrm{supCO}(L \setminus K, L, P)$ 也是 CO。根据 $L \setminus K = \overline{\{c\tau c, ca\tau\}}$, 令 $H_0 = H$, 如图 4.2 (右) 所示, 计算互补解。第一轮 ($i = 0$) 计算得到 $X_0^{CO} = X_0' = X_{0,m}' = \{1, 2, 3, 6, 7\}$, 从而得到 $\mathrm{supCO}(L \setminus K, L, P)$, 如图 4.3 所示。类似地, 根据命题 4.2, 有 $\mathrm{infCO}(K, L, P) = L \setminus \mathrm{supCO}(L \setminus K, L, P) = \{cc, c\tau c\}$, 这是通过交换关键和非关键状态得到的对偶解。

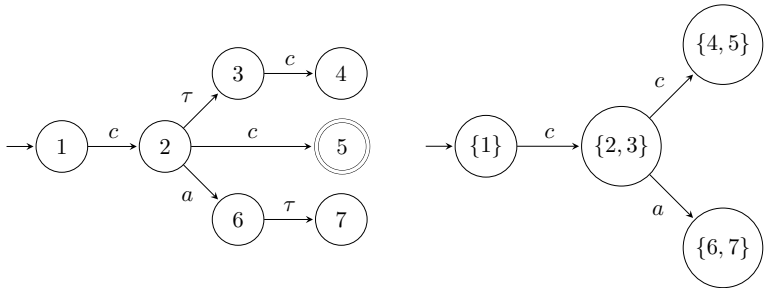

图 **4.1** 满足 $\Sigma_o = \Sigma_c = \{a, c\}$ 的 **DFA** G ； 观测器 $Obs(G)$

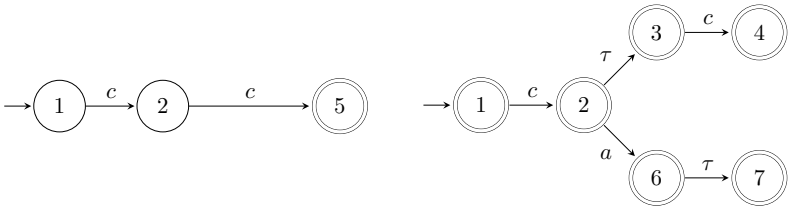

图 **4.2** 规范 K 和 $L \setminus K$ 对应的**DFA**

例 4.6. 接下来使用 G 和 $Q_c = \{5\}$ 来说明 $\mathrm{supCNCO}(K, L, P)$ 的计算, 其中 $L_m(H_0) = L_m(H) = K = \{cc\}$ 作为算法 4.1 的输入。第一轮 ($i = 0$) 计算规范条件：

$$X_0^C = \{1, 2, 5\}, X_0^N = \{1\}, X_0^{CO} = \{1, 2\},$$

② 符号 | 表示“限制于”。

③ 符号 $Trim(\cdot)$ 表示可访问部分和共可访问部分。

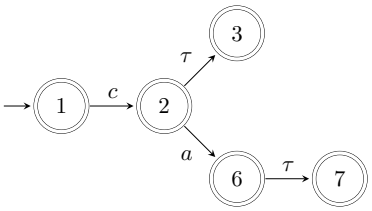

图 **4.3** $\mathrm{supCO}(L \setminus K, L, P)$ **(**也是 $\mathrm{supCNCO}(L \setminus K, L, P) = L_m(H_1)$**)**

推出 $L_m(H_2) = L_m(H_1) = \mathrm{supCNCO}(K, L, P) = \emptyset$.

现在, 根据命题 4.1, 可以选择从 $L_m(H_0) = L_m(H) = L \setminus K = \overline{\{c\tau c, ca\tau\}}$ 开始, 如图 4.2 (右) 所示, 计算对偶解。第一轮 ($i = 0$) 计算规范条件：

$$X_1^C = \{1, 2, 3, 4, 6, 7\}, X_1^N = \{1, 2, 3, 6, 7\}, X_1^{CO} = \{1, 2, 3, 6, 7\}.$$

步骤 4 和 5 得到 H_1, 如图 4.3 所示。第二轮 ($i = 1$) 得出 $H_2 = H_1$, 从而得到 $\mathrm{supCNCO}(L \setminus K, L, P)$, 如图 4.3 所示。

4.3 模块化确定性有限自动机的可控、常态和临界可观监督器的综合

本节为模块化系统 $L = \|_{i=1}^n L_i$ 使能临界可观测性。接下来的小节将分为两种情况介绍：针对局部规范和全局规范。

4.3.1 局部规范

首先考虑每个局部系统语言 L_i 都有其各自的关键语言 $K_i \subseteq L_i$ 的情况。例如, 如果 $\|_{i=1}^n G_i$ 的关键状态集合在局部由 $\times_{i=1}^n Q_{c,i}$ 给出, 其中 $Q_{c,i} \subseteq Q_i$, 那么可以得到相应的局部规范 K_i, 它们等于在状态 $Q_{c,i}$ 终止的字符串的集合。回顾上一节, 对于一个前缀闭合的系统语言 L 和一个规范 K, 其中 $K \subseteq L$, 将规范 K 相对于 L 和 P 的极大临界可观子语言表示为 $\mathrm{supCO}(K, L, P)$。

控制规范 K 可以是 $\mathrm{supCO}(K, L, P)$ 或 $\mathrm{supCO}(L \setminus K, L, P)$。这些语言中可能没有或只有一个是相对于字母表 $\Sigma_i, i = 1, \ldots, n$ 可分解的 (对于这些语言都不可分解的情况, 请参阅下一小节) 。

本节要解决的第一个问题是确定什么条件可以确保以下等式成立：

$$\mathrm{supCO}(K, L, P) = \|_{i=1}^n \mathrm{supCO}(K_i, L_i, P_{i,o}^i).$$

为简便起见, 当采用与 $L \setminus K$ 相应的对偶方法时, 将 $L \setminus K$ 简记为 K。假设 $L \setminus K = \|_{i=1}^{n} P_i(L \setminus K)$ 是可分解的 (否则需要下一小节的条件可分解性) , 且 $P_i(L \setminus K)$ 表示为 K_i。定义 2.1 的 MOC 对解决这个问题至关重要, 为了证明本节的主要结果, 利用以下引理。

引理 4.2. 考虑一个模块化的DFA $G = \|_{i=1}^{n} G_i$, 其中 $L = L(G)$ 和 $L_m = L_m(G)$。如果 L 对于 P_i、P 和 $P_{i,o}^{i}, i = 1, \ldots, n$ 是 MOC, 则$K \subseteq L_m$ 相对于 L 和 P 是 LCO 意味着 $P_i(K)$ 相对于 $P_i(L)$ 和 $P_{i,o}^{i}$ 也是 LCO。

证明. 上述表述等价于需要证明包含关系 $P_i(K) \supseteq (P_{i,o}^{i})^{-1}[P_{i,o}^{i}(P_i(K))] \cap P_i(L)$。取 $t' \in (P_{i,o}^{i})^{-1}[P_{i,o}^{i}(P_i(K))] \cap P_i(L)$。则存在 $s \in K \subseteq L$ 使得 $t' \in (P_{i,o}^{i})^{-1}[P_{i,o}^{i}(P_i(s))]$, 从而得到 $P_{i,o}^{i}(P_i(s)) = P_{i,o}^{i}(t')$。根据 MOC 条件, 存在 $s' \in L$ 使得 $P_i(s') = t'$ 且 $P(s) = P(s')$。根据 K 的 LCO 性质, $s' \in P^{-1}[P(s)] \cap L \subseteq P^{-1}[P(K)] \cap L = K$ 成立。因此$t' = P_i(s') \in P_i(K)$, 证毕。 □

定理 4.1. 考虑语言 $K_i \subseteq L_i \subseteq \Sigma_i^*$, 其中 L_i 是前缀闭合的, 对于 $i = 1, \ldots, n$ 且 $n \geq 2$。定义规范 $K = \|_{i=1}^{n} K_i$ 和 $L = \|_{i=1}^{n} L_i$。则以下结论成立：

1) $\mathrm{supCO}(K, L, P) \supseteq \|_{i=1}^{n} \mathrm{supCO}(K_i, L_i, P_{i,o}^{i})$。

2) 如果 $P_i(L) = L_i$ 且 L 对于 P_i、P 和 $P_{i,o}^{i}$ 是 MOC, 则

$$\mathrm{supCO}(K, L, P) = \|_{i=1}^{n} \mathrm{supCO}(K_i, L_i, P_{i,o}^{i}).$$

证明. 为了方便起见, 将 $\mathrm{supCO}(K_i, L_i, P_{i,o}^{i})$ 缩写为 supCO_i, 将 $\mathrm{supCO}(K, L, P)$ 缩写为 supCO。

为了建立 1), 需要证明 $\|_{i=1}^{n}\ \mathrm{supCO}_i$ 相对于 L 和 P 是 LCO, 因为这意味着 $\|_{i=1}^{n} \mathrm{supCO}_i$ 包含在 $\mathrm{supCO}(K, L, P)$ 中。更准确地说, 可以推出以下包含关系

$$\begin{aligned}
\|_{i=1}^{n} \mathrm{supCO}_i &\subseteq P^{-1}[P(\|_{i=1}^{n}\ \mathrm{supCO}_i)] \cap L \\
&\subseteq P^{-1}[(\|_{i=1}^{n} P_{i,o}^{i}(\mathrm{supCO}_i)] \cap L \\
&= \|_{i=1}^{n} (P_{i,o}^{i})^{-1}[P_{i,o}^{i}(\mathrm{supCO}_i)] \cap L \\
&= \|_{i=1}^{n} (P_{i,o}^{i})^{-1}[P_{i,o}^{i}(\mathrm{supCO}_i)] \cap \|_{i=1}^{n} L_i \\
&= \|_{i=1}^{n} [(P_{i,o}^{i})^{-1}[P_{i,o}^{i}(\mathrm{supCO}_i)] \cap L_i] \\
&= \|_{i=1}^{n} \mathrm{supCO}_i,
\end{aligned}$$

这证明了 $\|_{i=1}^{n}\mathrm{supCO}_i$ 是 LCO.

为了建立 2), 条件 $P_i(L)=L_i$ 和 L 是 MOC, 以及引理 4.2, 一同表明 $P_i(\mathrm{supCO})$ 相对于 $P_i(L)=L_i$ 和 $P_{i,o}^{i}$ 是 LCO。因此, 对于 $i=1,\ldots,n$, 有 $P_i(\mathrm{supCO})\subseteq \mathrm{supCO}_i$, 从而 $\mathrm{supCO}\subseteq\|_{i=1}^{n}\mathrm{supCO}_i$。 □

如下更直观的结果它可以直接从引理 2.1 和定理 4.1 中得出, 故证明略去。

定理 4.2. 设 $L=\|_{i=1}^{n}L_i$ 为由前缀闭合语言 L_i 组成的模块化系统, 其中 L_i 定义在字母表 Σ_i 上, 对于 $i=1,\ldots,n$ 且 $n\geq 2$。设 $K=\|_{i=1}^{n}K_i$, 其中 $K_i\subseteq L_i$ 是规范。如果 $P_i(L)=L_i$ 且 $\Sigma_s\subseteq\Sigma_o$, 则

$$\mathrm{supCO}(K,L,P)=\|_{i=1}^{n}\mathrm{supCO}(K_i,L_i,P_{i,o}^{i}). \qquad \square$$

上述定理保证了如果算法从一个可分解的关键语言作为规范 (或可分解的非关键语言) , 那么 K (或 $L\setminus K$) 的极大 LCO 子语言将保持可分解。这对于在应用定理 4.3 之后的模块化综合过程非常重要, 因为计算可以从一个可分解的 LCO 语言开始, 这意味着其局部规范将作为算法 1 中的规范。

现在回到通过监督控制综合 $\mathrm{supCO}(K,L,P)$ 的问题。回顾一下, 算法 4.1 计算了极大可控、常态和临界可观语言 $\mathrm{supCNCO}(K,L,P)$, 这可以被视为在所有可控事件都可观的情况下使能临界可观测性的最大许可监督器。一个重要的问题是, 在什么条件下 $\mathrm{supCNCO}(K,L,P)$ 可以被模块化监督器 $\mathrm{supCNCO}(K_i,L_i,P_{i,o}^{i})$, $(i=1,\ldots,n)\wedge(n\geq 2)$ 替代。

请注意, 为了将CO方面的最大许可性 (本文中的定理 4.2) 和常态性方面的最大许可性 ([43] 中的定理 8) 与可控性方面的最大许可性结合, 以下定理使用了观测器和局部控制一致性 (Local Control Consistency, LCC) 属性, 参见 [127]。

定理 4.3. 对于一个模块化的DFA $G=\|_{i=1}^{n}G_i$, 其中 $L=L(G)=\|_{i=1}^{n}L_i=\|_{i=1}^{n}L(G_i)$, 以及一个可分解的规范 $K=\|_{i=1}^{n}K_i$, 其中 $n\geq 2$, 如果对于 $i=1,\ldots,n$,

1) $\mathrm{supCNCO}(K_i,L_i,P_{i,o}^{i})$ 不互相冲突,

2) $P_i(L)=L_i$,

3) $\Sigma_s\subseteq\Sigma_o$,

4) P_i 是 L 的观测器,

5) P_i对于L是局部控制一致的,

则

$$\mathrm{supCNCO}(K, L, P) = \|_{i=1}^{n} \mathrm{supCNCO}(K_i, L_i, P_{i,o}^i).$$

证明. 证明可以通过将本文中的定理 4.2 (关于 CO) 与 [43] 中的定理 16 (关于可控性和常态性) 结合起来获得。 □

另一种方法是, 假设 4) 和 5) 可以被所有公有事件都是可控的假设 (即 $\Sigma_s \subseteq \Sigma_c$) 来替代, 这类似于 [43] 中的推论 17。考虑以下示例。

例 4.7. 考虑一个由两个模块 G_1 和 G_2 组成的模块化系统, 其中 $G_1 = G$ 如图 4.1 所示, G_2 如图 4.4 所示。令 $L_i = L(G_i)$, $i = 1, 2$, 且 $L = L_1 \parallel L_2$。注意, $\mathrm{supCNCO}(L \setminus K_1, L_1, P_{1,o}^1)$ 已在示例 4.6 中计算, 并且如图 4.3 所示。使用算法 4.1 可以得到 $\mathrm{supCNCO}(L \setminus K_2, L_2, P_{2,o}^2)$, 如图 4.5 所示。可观事件的集合是 $\Sigma_o = \{a, b, c\}$。所有公有事件 ($\Sigma_s = \{c\}$) 都是可观和可控的。因为 $P_i(L) = L_i$, 且 $\mathrm{supCNCO}(L \setminus K_i, L_i, P_{i,o}^i)$ 是前缀闭合的, 所以它们也是非冲突的。这些结论使得定理 4.3 适用于这个示例。因此, 有

$$\mathrm{supCNCO}\big((L \setminus K_1) \parallel (L \setminus K_2), L_1 \parallel L_2, P\big) = \mathrm{supCNCO}(L \setminus K_1, L_1, P_{1,o}^1) \parallel \mathrm{supCNCO}(L \setminus K_2, L_2, P_{2,o}^2).$$

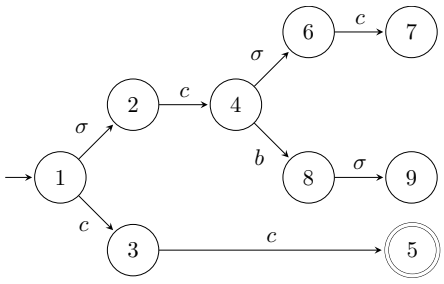

图 **4.4 DFA** G_2**,** 其中 $\Sigma_o = \Sigma_c = \{b, c\}$

请注意, 与定理 4.1 和定理 4.2 中的极大临界可观子语言的模块化计算不同, 在这些定理中对局部 $\mathrm{supCO}(K_i, L_i, P_{i,o}^i)$ 的非冲突性没有要求, 而在定理 4.3 中, 这种非冲突性是必要的。这种必要性出现是因为, 与可控性和常态性不同, LCO 是定义在语言自身, 而不是其前缀闭包的性质。冲突问题是模块化监督控制中的一个已知困难, 文献中已经提出了几种解决方案, 见 [127-130]。一种可能的解决方案是使用适当字母表上的协调器形式的抽象, 以确保这些字母表的观测器属性, 如 [131] 中所示。

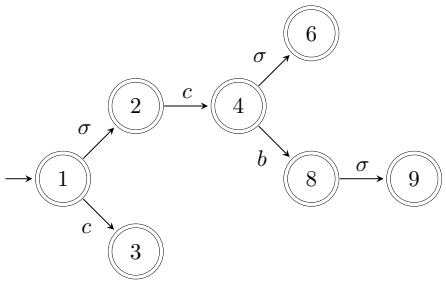

图 **4.5** $\mathrm{supCNCO}(K_2, L_2, P^2_{2,o})$

4.3.2　全局规范

本小节考虑不可分解的关键语言的情况。主要方法基于协调控制框架中的条件可分解性概念 (参见下文) [131]。如果 K 不是可分解的, 可以通过在局部组件之间通信一些协调事件 (记作 Σ_κ) 来扩展公有字母表。

文献 [131] 的结论表示, 总是存在一个字母表 $\Sigma_\kappa \subseteq \Sigma$, 使得 K 是条件可分解的[132], 即相对于更大的字母表 $\Sigma_{i+\kappa} = \Sigma_i \cup \Sigma_\kappa$ 是可分解的。在确定了字母表 Σ_κ 后, 协调器 G_κ 可以以分布式的方式计算, 其定义为 DFA, 满足

$$L_\kappa = L(G_\kappa) = P_\kappa(L) = \|^n_{i=1} P^i_{\kappa,i}(L_i),$$

其中 $P_\kappa : \Sigma^* \to \Sigma^*_\kappa$ 和 $P^i_{\kappa,i} : \Sigma^*_i \to \Sigma^*_{i,\kappa}$。

这种转换使得原始的模块化系统转换为新的模块化系统, 其中模块为

$$L_{i+\kappa} = P_{i+\kappa}(L) = L_i \parallel L_\kappa,$$

公有事件集为 Σ_κ, 并且局部规范为

$$K_{i+\kappa} = P_{i+\kappa}(K).$$

这推导出了一个系统语言 $L = \|^n_{i=1} L_i = \|^n_{i=1} L_{i+\kappa}$ 和一个规范 $K = \|^n_{i=1} K_{i+\kappa}$。

很容易看出, 极大临界可观子语言的模块化计算可以从引理 2.1 和定理 4.1 中直接推导出来。请注意, 当使用协调器时, 总是有 $P_{i+\kappa}(L) = L_{i+\kappa}$。如果所有公有事件都是可观的, 即 $\Sigma_\kappa \subseteq \Sigma_o$, 则

$$\mathrm{supCO}(K, L, P) = \|^n_{i=1} \mathrm{supCO}(K_{i+\kappa}, L_{i+\kappa}, P^{i+\kappa}_{i+\kappa,o}),$$

其中 $P^{i+\kappa}_{i+\kappa,o} : \Sigma^*_{i+\kappa} \to \Sigma^*_{i+\kappa,o}$。

同样, 定理 4.3 也可以扩展到不可分解规范的情况。值得注意的是, 在模块化

综合中, 仅需基于算法 1 计算 $\mathrm{supCNCO}(K_{i+\kappa}, L_{i+\kappa}, P^{i+\kappa}_{i+\kappa,o})$。

4.4 本章小结

本章引入了基于语言的临界可观测性, 并证明了它在DFA框架下与基于状态的临界可观测性是等价的。一般来说, 常态性和临界可观测性不可比较, 因为常态性涉及语言的前缀闭包, 而临界可观测性则不涉及。此外, 本章提出了一种计算极大可控、常态且临界可观子语言的算法, 适用于一般 (非前缀闭合的) 规范。基于常态性与临界可观测性之间的相似性, 本章应用了 MOC 条件来进行临界可观监督器的模块化综合, 并考虑了局部和全局规范。下一章将研究离散事件系统的故障诊断与预测, 填补现有文献中关于相关问题复杂度分析的空白, 并为后续恒定时间自动机及时间区间自动机上的故障诊断及控制的研究奠定基础。

第5章　弱可诊断性和弱可预测性的问题可判定性

5.1　判定有限状态自动机的弱可诊断性和弱可预测性

本节主要研究NFA弱可诊断性和弱可预测性的可判定性, 证明弱诊断性和A-诊断性的概念是一致的, 并证明判定弱预测性是PSPACE-complete和判定弱模块预测性是EXPSPACE-complete。

本章做出以下标准假设, 因为本文并不关注系统在未来可能缺乏可观行为的情况, 这与文献中的做法相同。

假设 5.1. 系统 G 中不存在死锁状态。

假设 5.2. 在 G 中不存在关于不可观事件集 $\Sigma_{uo} = \Sigma \setminus \Sigma_o$ 的不可观循环。

5.1.1　弱可诊断性的计算复杂度

弱可诊断性最初由文献 [63] 提出, 其规定每个故障都必须有可能被诊断出来。这一概念相比文献 [48] 中提出的可诊断性要求 (每个发生的故障必须在有限步内被诊断出) 更为宽松, 因此在实际系统中所需的传感器较少。首先回顾NFA 中弱可诊断性的定义。

故障诊断相关问题通常用 $\Sigma_f \subseteq \Sigma_{uo}$ 表示故障事件的集合, 这在后续两章中同理。设 $\Psi(\Sigma_f) = \{we_f \in L(G) \mid e_f \in \Sigma_f\}$ 为 $L(G)$ 中以故障事件 $e_f \in \Sigma_f$ 结尾的字符串的集合。稍作符号滥用, 符号 $\Sigma_f \in w$ 表示存在 $e_f \in \Sigma_f$, 使得 e_f 出现在字符串 w 中。

定义 5.1 (NFA的弱可诊断性[63])**.** 一个NFA $G = (Q, \Sigma, \delta, I)$, 如果对于任何包含故障事件的字符串, 都存在一个字符串扩展使得故障能够被检测到, 则称其关于投影 $P : \Sigma \to \Sigma_o$ 和故障事件 Σ_f 是**弱可诊断的**, 即

$$(\forall w_1 \in \Psi(\Sigma_f))(\forall w_2 \in L/w_1)(\exists w_3 \in L/w_1w_2)(\forall w \in L(G)) \\ [P(w) = P(w_1w_2w_3) \Rightarrow (\Sigma_f \in w)].$$

在文献 [54] 中, 提出了A- (Almost-)可诊断性 (最初在文献 [133] 中为随机自动机定义) 的一个等价的逻辑版本。下文将展示A-可诊断性与弱可诊断性是等价的。因此, 关于A-可诊断性的现有结果可以应用于弱可诊断性。

定义 5.2 (NFA的A-可诊断性[54]). 设 $L_f = \Sigma^*\Sigma_f\Sigma^*$ 为所有包含故障的字符串的集合。对于给定的投影 $P : \Sigma \to \Sigma_o$ 和故障事件集 Σ_f, 如果NFA G 满足

$$(\forall s \in L(G) \cap L_f)(\exists t \in L/s)(\forall w \in L(G))[P(w) = P(st) \Rightarrow (\Sigma_f \in w)],$$

则称 G 是A-可诊断的。

命题 5.1. NFA G 是弱可诊断的当且仅当它是 A-可诊断的。

证明. 注意, $w_1 \in \Psi(\Sigma_f) \iff w_1 \in \Sigma^*\Sigma_f$ 成立。这意味着, 对于 $w_1 \in \Psi(\Sigma_f)$, w_2 是 w_1 在 L 中的扩展当且仅当 $w_1w_2 \in L(G) \cap \Sigma^*\Sigma_f\Sigma$。令 $t = w_3$, 证毕。 □

综上所述, 结合文献 [134-135] 中的结果, 通过将A-可诊断性判定问题转化为全局性问题 (Universality Problem) , 证明了验证NFA的A-可诊断性是PSPACE-complete, 因此判定弱可诊断性也是 PSPACE-complete。

5.1.2 弱可预测性的计算复杂度

本小节介绍了离散事件系统基于观测来预测部分故障的能力 (称为弱可预测性) , 并使用 NFA 研究其可判定性。首先回顾可预测性的定义。

定义 5.3 (NFA的可预测性[52]). 对于 P 和 Σ_f, NFA G 是可预测的, 如果

$$(\exists K \in \mathbb{N})(\forall w_1 \in \Psi(\Sigma_f))(\exists w_2 \in \overline{w_1})[\Sigma_f \notin w_2 \wedge \mathbf{P}],$$

其中, 预测性条件 $\mathbf{P}$ 是

$$(\forall w \in L(G))(\forall w' \in L/w)[(P(w) = P(w_2)) \wedge (\Sigma_f \notin w) \wedge (|w'| \geq K) \Rightarrow (\Sigma_f \in w')].$$

下面提出弱可预测性的定义。

定义 5.4 (NFA的弱可预测性). 对于 P 和 Σ_f, NFA G 是**弱可预测的**, 如果

$$(\exists K \in \mathbb{N})(\exists w_1 \in \Psi(\Sigma_f))(\exists w_2 \in \overline{w_1})[\Sigma_f \notin w_2 \wedge \mathbf{P}].$$

简单来说, 弱可预测性是一个新的概念, 要求系统能够提前几步预测某些故障的发生。与此相比, 可预测性要求系统能够提前几步预测每一个故障的发生。下面提供了一个例子来说明这种区别。

例 5.1. 考虑图 5.1 中所示的NFA G。显然, 这个系统是弱可预测的。这是因为, 对于故障事件 e_f, 在观测到 a 后, e_f 将在一步内发生是可以预测到的。然而, 当观测到 b 时, e_f 是否会发生无法预测, 这表明 G 不是可预测的。

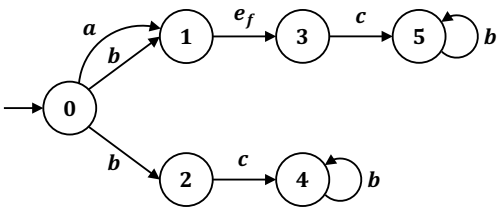

图 **5.1 NFA** $G = (Q, \Sigma, \delta, I)$**,** 其中 $\Sigma_o = \{a, b, c\}$ 且 $\Sigma_f = \{e_f\}$

值得注意的是, 弱可预测性和弱可诊断性是不可比较的。换句话说, 弱可预测性并不意味着弱可诊断性, 反之亦然, 如下面的例子所示。这一点与可预测性和可诊断性之间的关系明显不同, 在文献 [52] 中指出了可预测性意味着可诊断性。

例 5.2. 再次考虑图 5.1 中的 G。系统 G 不是弱可诊断的, 因为对于字符串 be_f, 无论在未来任何状态下, e_f 的发生都无法检测到。另一方面, 如果移除变迁 $0 \xrightarrow{a} 1$ 并将 $2 \xrightarrow{c} 4$ 替换为 $2 \xrightarrow{a} 4$, 则得到的自动机变为弱可诊断的但不是弱可预测的。

将文献 [52] 中用于通过诊断器验证可预测性的结果拓展用于验证弱可预测性。给定一个NFA G, 其**诊断器** (Diagnoser) 记作

$$Diag(G) = Obs(G \parallel A_\ell) = (X, \Sigma_o, \xi, x_0) = Ac(2^{Q\times\{N,F\}}, \Sigma_o, \xi, x_0),$$

其中 $A_\ell = (\{N, F\}, \Sigma_f, \delta_\ell, \{N\})$ 是标记自动机, 满足 $\delta_\ell(N, e_f) = \{F\}$ 和 $\delta_\ell(F, e_f) = \{F\}$ 对于所有 $e_f \in \Sigma_f$。变迁函数 $\xi : X \times \Sigma_o \to X$ 定义为 $\xi(x, a) = \{q \in Q \times \{N, F\} \mid \exists q' \in x, \exists v \in \Sigma_{uo}^* : q \in \delta_{G\|A_\ell}(q', va)\}$, 其中 $\delta_{G\|A_\ell}$ 是 $G\|A_\ell$ 的变迁函数。变迁函数 ξ 可以按常规方法拓展至 $X \times \Sigma_o^*$。

一个状态 $x = \{(q_1, \ell_1), (q_2, \ell_2), \ldots, (q_m, \ell_m)\} \in X$, 对于 $m \in \mathbb{N}$, 被定义为:

- **安全的**, 如果 $\ell_1 = \ell_2 = \cdots = \ell_m = N$;
- **故障的**, 如果 $\ell_1 = \ell_2 = \cdots = \ell_m = F$;
- **模糊的**, 如果存在某些 $j, k \in \{1, 2, \ldots, m\}$ 使得 $\ell_j = N$ 且 $\ell_k = F$。

令 $X_N \subseteq X$、$X_F \subseteq X$ 和 $X_U \subseteq X$ 分别表示安全状态、故障状态和模糊状态的集合。定义 $X_{\neg N}^N \subseteq X$ 为安全状态中具有指向故障或模糊状态的输出变迁的状

态集合。形式上, $X^N_{\neg N} = \{x \in X_N \mid \exists a \in \Sigma_o : \xi(x,a) \in X_F \cup X_U\}$。此外, 使用 $Ac(Diag(G),x) = (X_{ac}, \Sigma_o, \xi_{ac}, x)$ 来表示在 $Diag(G)$ 中状态 $x \in X$ 的可达部分, 其中 $X_{ac} = \{x' \in X \mid \exists w \in \Sigma_o^* : \xi(x,w) = x'\}$, 并且 $\xi_{ac} = \xi \mid X_{ac}$ 限制在 X_{ac} 上。

在文献 [52] 中, 验证可预测性的充要条件被表述如下: 一个NFA G 是可预测的, 当且仅当对于每个 $x \in X^N_{\neg N}$, $Ac(Diag(G),x)$ 中的所有循环仅由故障状态组成。请注意, 对于每个状态 $x = \{(q_1,N),(q_2,N),\dots\} \in X^N_{\neg N}$, 存在 (q_i,N) 使得故障事件 e_f 在 q_i 上是可行的, 即 $\delta(q_i,e_f) \subseteq Q$。因此, 结合定义 5.3 和 5.4, 可以得出以下引理。证明可以很容易地根据文献 [52] 中定理8的证明进行调整, 因此省略。

引理 5.1. 一个NFA G 是弱可预测的, 当且仅当存在 $x \in X^N_{\neg N}$, 使得 $Ac(Diag(G),x)$ 中的所有循环仅由故障状态组成。

例 5.3. 重新考虑图 5.1 中的 NFA G。它的诊断器 $Diag(G)$ 如图 5.2 所示。根据定义, 有 $X^N_{\neg N} = \{\{(1,N)\},\{(1,N),(2,N)\}\}$。由于对于 $\{(1,N)\}$, 存在一个唯一的故障循环 $\{(5,F)\} \xrightarrow{b} \{(5,F)\}$, 根据引理 5.1, G 是弱可预测的。

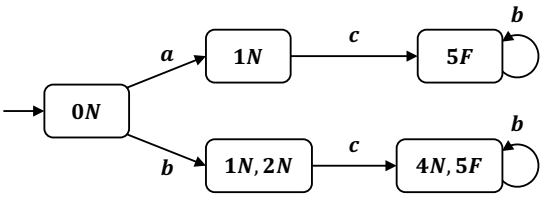

图 **5.2** 诊断器 $Diag(G)$

直观上, 人们可能会认为判定弱可预测性比判定弱可诊断性是一个更复杂的问题。然而, 以下证明显示, 从复杂度理论的角度来看, 在NFA中, 判定弱可预测性并不会本质上比判定弱可诊断性更难处理。

定理 5.1. 判定一个NFA是否弱可预测是 PSPACE-complete。

证明. “成员性”: 请注意, 诊断器 $Diag(G)$ 的每个状态都是 $Q \times \{N,F\}$ 的一个子集, 因此其大小是原NFA G 状态数量的多项式函数。因此, 通过在 $Diag(G)$ 中使用非确定性搜索来检验引理 5.1, 该算法在 PSPACE 中运行。

“难度”: 以下证明将正则表达式的包含问题[136] (这是 PSPACE-complete 问题) 归约到弱可预测性决策问题。

设 $G_i = (Q_i, \{\clubsuit, \diamondsuit\}, \delta_i, \{q_0^i\})$, 其中 $i = 1, 2$ 是两个 NFA, 其中 $Q_1 \cap Q_2 = \emptyset$。设 $F_1 \subseteq Q_1$ 是 G_1 中的最终状态集合。G_1 接受的语言为 $L_m(G_1) = \{w \in \{\clubsuit, \diamondsuit\}^* \mid \delta_1(q_0^1, w) \cap F_1 \neq \emptyset\}$。设 $G = (Q_1 \cup Q_2 \cup \{q_0, q_f\}, \{\clubsuit, \diamondsuit, e_f\}, \delta, \{q_0\})$, 其中 q_0 和 q_f 是不在 $Q_1 \cup Q_2$ 中的新的状态, e_f 是唯一的不可观事件, 且 δ 定义如下: 对于 $(p, x, q) \in \delta_i$, 其中 $i = 1, 2$, 有 $(p, x, q) \in \delta$。对于 $q \in F_1$, 令 $(q, e_f, q_f) \in \delta$。对于 $(q_0, \diamondsuit, q_0^1)$ 和 $(q_0, \diamondsuit, q_0^2)$, 有 $(q_0, \diamondsuit, q_0^1), (q_0, \diamondsuit, q_0^2) \in \delta$。得到的 G 是如图 5.3 所示的 NFA。

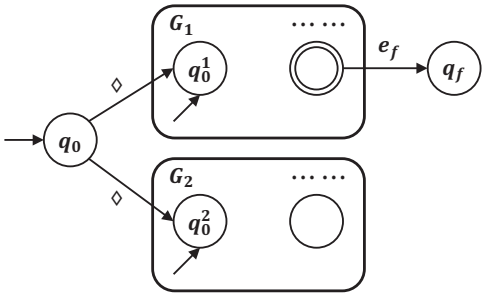

图 **5.3** 定理 **5.1** 归约证明的示意图

现在证明 $L_m(G_1) \subseteq L(G_2)$ 当且仅当 G 不是弱可预测的。如果 $L(G_1) \subseteq L(G_2)$, 那么对于每个字符串 $\diamondsuit w e_f$, 其中 $w \in L_m(G_1) \subseteq L(G_2)$, 总是可以找到一个字符串 $\diamondsuit w$ 具有相同的投影。因此, 永远无法预测是否会发生故障, 这导致 G 不是弱可预测的。如果 $w \in L_m(G_1) \wedge w \notin L(G_2)$, 那么在观测到 w 后, e_f 将在有限的步数内发生。因此, G 是弱可预测的, 证毕。 □

在上述构造中, 至少一个不可观事件是不可避免的, 因为故障事件总是不可观的。此外, 至少需要两个可观事件, 因为针对单一字母表的 NFA 的语言包含问题 (Language Inclusion Problem) 是 coNP-complete[137]。

5.1.3　弱模块化可预测性的计算复杂度

给定一组 NFA $G_1, \ldots, G_n$、一组不可观事件 Σ_{uo} 和一个故障事件 $e_f \in \Sigma_{uo}$, 弱模块化可诊断性 (或弱模块化可预测性) 问题是询问系统 $\|_{i=1}^n G_i$ 是否关于 Σ_{uo} 和 e_f 是弱可诊断的 (或弱可预测的) 。

如上所述, A-可诊断性实际上等价于弱可诊断性。在文献 [54] 中, 判定模块化 NFA 的 A-可诊断性被证明为 EXPSPACE-complete 问题。接下来, 通过修改文献 [54] 中判定模块化A-可诊断性的证明, 以证明判定弱模块化可预测性也是

EXPSPACE-complete 问题。

定理 5.2. 判定弱模块化可预测性是 EXPSPACE-complete。

证明. “成员性”：注意到 $Obs(G = \|_{i=1}^{n} G_i)$ 的状态是 n 元组的子集, 因此每个状态的基数最多为 k^n, 其中 k 是所有 G_i 中状态数量的最大值。然后, 通过使用非确定性搜索 (参见定理 5.1 的证明) 方法可以证明该问题属于 EXPSPACE 类。

“难度”：将包含平方算子的正则表达式的全局性问题 (已知为 EXPSPACE-complete) 进行归约。包含平方的正规表达式可以涉及操作 $\cdot$、$*$、$\cup$ 和平方运算 $w^2 = w \cdot w$。在文献 [136] 中, 已经证明只需考虑字母表 Δ 上的表达式 E, 形式如 [136] 的引理 2.1 证明中所示。对应的 (模块化) NFA G_i (其中 $i = 1, \ldots, n$ 且 $n = |\Delta|^3 + 5$) 的构造方法在 [54] 的定理 2 的证明中给出。定义 $\|_{i=1}^{n} G_i$ 的字母表为 Σ, 构造表明 $L(E) = \bigcup_{i=1}^{n} P\big(L_m(G_i)\big)$, 其中 $P : \Sigma \to \Delta \cup \{\diamondsuit\}$, 且 $\Delta \cup \{\diamondsuit\} \subseteq \Sigma$, $\diamondsuit$ 是一个新增的可观事件, 使用方式如下: 对于每个 G_i, 引入两个新状态, q_{s_i} 和 q_{f_i}。如果 w 被 $P(G_i)$[①] 接受, 则 q_{s_i} 和 q_{f_i} 都可以通过 $w\diamondsuit$ 在 $P(G_i)$ 中到达。相反, 如果 w 不被 $P(G_i)$ 接受, 则只有 q_{s_i} 可以通过 $w\diamondsuit$ 在 $P(G_i)$ 中到达。关于该构造的更多细节, 请参阅 [54]。图 5.4 提供了概念化的示意图。

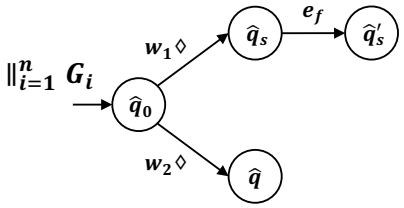

图 **5.4** 定理 **5.2** 归约证明的概念化草图

现在证明, 当且仅当$\|_{i=1}^{n} G_i$ 关于 $\Sigma \setminus (\Delta \cup \{\diamondsuit\})$ 和 $\{e_f\}$ 是弱可预测性的, 则 $L(E) \neq \Delta^*$。若 $L(E) = \Delta^*$, 对于每一个 $w \in \Delta^*$, 在观测到 $w\diamondsuit$ 后, 系统 $\|_{i=1}^{n} G_i$ 可能处于至少两个状态, 即 $\hat{q}_s = (q_{s_1}, q_{s_2}, \ldots, q_{s_n})$ 和 $\hat{q} = (\ldots, q_{f_i}, \ldots)$, 其中“$\ldots$”表示 q_{f_j} 或 q_{s_j}, 且 $j \in \{1, 2, \ldots, n\}$ 及 $j \neq i$。这意味着 e_f 的发生无法预测, 从而导致 $\|_{i=1}^{n} G_i$ 不是弱可预测的。反之, 如果 $L(E) \neq \Delta^*$, 则存在 $w \in \Delta^*$ 使得 $w \notin L(E) = \bigcup_{i=1}^{n} P(L_m(G_i))$。因此, 在观测到 w 之后, 没有一个 G_i 处于标记状态。由于事件 $\diamondsuit$ 将 G_i 的每个非标记状态仅引向 q_{s_i}, 因此字符串 $w\diamondsuit$ 仅将 $\|_{i=1}^{n} G_i$ 引向状态 $\hat{q}_s$。这意味着在观测到 $w\diamondsuit$ 后, 可以预测到故障 e_f 将在有限步数内发生。因此, $\|_{i=1}^{n} G_i$ 是弱可预测性的。 □

① 符号 $P(G)$ 表示带有 ε-变迁的 NFA, 即 $P(G) = (Q, \Sigma_o, \delta', I)$, 其中 $\delta' = \{(p, P(a), q) \mid (p, a, q) \in \delta\}$。

如果所有不可观事件都是私有的, 那么投影和并行组合运算可以被交换, 即 $P(L_m(\|_{i=1}^{n} G_i)) = \|_{i=1}^{n} P(L_m(G_i))$。通过这种方式, 判定弱模块化可预测性的复杂度可以降低到 PSPACE-complete。

定理 5.3. 给定一个模块化NFA $\|_{i=1}^{n} G_i$, 以及一个投影 $P : \Sigma \to \Sigma_o$, 使得任意两个NFA的所有公有事件都属于 Σ_o, 则判定弱模块化可预测性是PSPACE-complete。

证明. 回顾 [54] 中的引理 3, 它表明如果任意两个NFA的所有公有事件都是可观的, 则状态在 $P(\|_{i=1}^{n} G_i)$ 中可通过一个字符串 $P(w)$ 到达, 当且仅当它在 $\|_{i=1}^{n} P(G_i)$ 中通过 $P(w)$ 可达。注意到故障事件是不可观的, 因此是私有的, 即每个故障只能在局部发生。令 $Diag(P(G_i))$ 表示 $P(G_i)$ 的确定化。通过在 $\|_{i=1}^{n} Diag(P(G_i))$ 应用引理 5.1, 说明问题是 PSPACE 的。结合定理 5.1 的推导的 PSPACE-hard。因此, 判定弱模块化可预测性是 PSPACE-complete。 □

5.2 判定标签Petri网的弱可诊断性和弱可预测性

由于PN语言 (由LPN表示) 相比于正则语言 (由NFA表示) 具有更大的语言表达能力, 因此将对NFA获得的结果扩展到LPN是有意义的。然而, 本节将展示, 对于无界 (Unbounded) LPN, 弱可诊断性和弱可预测性都是不可判定的。

5.2.1 弱可诊断性的可判定性

设 $N = (P, T, Pre, Post)$, 并令 $T_f \subseteq T$ 表示故障变迁的集合。用 $\Psi(T_f) = \{\sigma t_f \in L(N, M_0) \mid t_f \in T_f\}$ 表示 $L(N, M_0)$ 中以故障变迁 $t_f \in T_f$ 结尾的序列集合。特别地, 对于一个序列 $\sigma = t_1 t_2 \cdots t_n \in T^*$, 用 $T_f \in \sigma$ 来表示序列 σ 中出现了故障变迁, 这意味着 $\exists i \in \{1, \ldots, n\} : t_i \in T_f$。

定义 5.5 (LPN的弱可检测性)**.** 一个LPN系统 $G = (N, M_0, \Sigma, \ell)$ 对于故障变迁 T_f 是弱可诊断的如果

$$(\forall s \in \Psi(T_f))(\forall s' \in L(N, M_0)/s)(\exists s'' \in L(N, M_0)/ss')$$
$$(\forall \sigma \in L(N, M_0))[\ell(\sigma) = \ell(ss's'') \Rightarrow (T_f \in \sigma)].$$

以下示例说明了 LPN 中弱可诊断性的概念。

例 5.4. 考虑图 5.5 中展示的一个LPN。直观上, 这个系统是弱可诊断的, 因为对于任何变迁序列 $s = t_f^{k_1}$ 及其观测 ε, 以及任何后续序列 $s' = t_f^{k_2} t_1^{k_3}$ 及其观测 a^{k_2}, 其中

$k_1, k_2, k_3 \in \mathbb{N}$, 总是存在一个进一步的后续序列 $s'' = t_1$ 及其观测 a, 使得故障可以被检测到。换句话说, 一旦观测到标签 a, 可以得出故障已经发生的结论。然而, 该系统不是可诊断的, 因为在故障变迁 t_f 发生之后, 系统可能会无限期地停留在标记 $[0, 1, 0]$ 中。在这种情况下, 故障将永远无法被检测到。

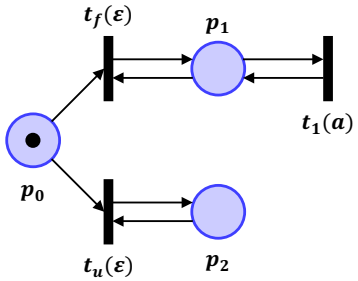

图 **5.5** **LPN** (N, M_0, Σ, ℓ)**,** 其中 $T_o = \{t_1\}$**,** $T_{uo} = \{t_u, t_f\}$**,** $\Sigma = \{a\}$**,** $\ell(t_1) = a$**,** 且 $\ell(t_u) = \ell(t_f) = \varepsilon$

尽管 LPN 的弱可诊断性是否可判定仍是一个开放问题, 但下面的结果表明, 如果它是可判定的, 它至少与可达性问题一样困难, 可达性问题已知是 Ackermann-complete[138-139]。可达性问题询问, 给定一个 PN $G = (N, M)$ 和一个标记 M', 标记 M' 是否从初始标记 M 在 G 中可达?

定理 5.4. 对于无界LPN, 验证弱可诊断性的问题是 Ackermann-hard。

证明. 将可达性问题归约为弱可诊断性问题。给定一个 LPN 系统 (N, M_0, Σ, ℓ), 其中 $N = (P, T, Pre, Post)$, 其中 $P = \{p_1, \ldots, p_n\}$, 以及一个标记 M, 添加

- 一个新的库所 $p_a \notin P$, 初始化时没有令牌,
- 一个新的变迁 $t_a \notin T$, 其中 $\ell(t_a) = a \notin \Sigma$,
- 一个新的故障变迁 $t_f \notin T$, 其中 $\ell(t_f) = \varepsilon$, 以及
- 一个新的变迁 $t_{M_i} \notin T$, 其中 $\ell(t_{M_i}) = \varepsilon, i = 1, \ldots, n$。

定义变迁 t_f 在标记 M 才能触发, 即对于每个库所 $p \in P$, $Pre(p, t_f) = M(p)$, 并且 $Pre(p_a, t_f) = 0$。因此, t_f 可以在覆盖 M 的每个标记中触发。触发 t_f 会放置一个令牌到新库所 p_a, 即对于 $p \in P$, $Post(p, t_f) = 0$, 并且对于 $p = p_a$, $Post(p, t_f) = 1$。此外, 在新库所 p_a, 仅在 a 下启用自循环, 即定义 $Pre(p_a, t_a) = Post(p_a, t_a) = 1$; 否则, 它是未定义的。请注意, 如果变迁 t_f 在标记 M 中触发, 则会导致在标记中仅 p_a 里有一个令牌, 而在 P 的库所没有令牌。特别地, 从现在开始只能观测到 a^*。

变迁 t_{M_i} 在每个标记中都可用, 这些标记在库所 p_i 至少包含 $M(p_i)+1$ 个令牌, 在库所 p_j 至少包含 $M(p_j)$ 个令牌, 其中 $j \neq i$; 也就是说, 对于每个严格覆盖 M 的标记 M', 都有一个变迁 t_{M_k} 可以触发, 但变迁 t_{M_i} 中的任何一个都不能在标记 M 触发。具体来说, 对于 $i=1,\ldots,n$, 定义 $Pre(p_i,t_{M_i})=M(p_i)+1$、$Pre(p_j,t_{M_j})=M(p_j)$、$j\neq i$、$Post(p_i,t_{M_i})=0$ 和 $Post(p_a,t_{M_i})=1$。同样, 在变迁 t_{M_i} 触发后, 网将到达一个标记, 其中 a 下的自循环可启用; 但是, 这一次网络 N 中可能仍有变迁可以启用, 因为原始网络 N 的 P 库所上仍有令牌。

请注意, 在覆盖 M 的每个标记中, t_f 和部分 t_{M_i} 是可启用的。因此, 如果 M 无法从 M_0 到达 G, 则对于某些 t_{M_i}, 包含故障变迁 t_f 的每个变迁序列 $w_1t_fw_2$ 都具有等价可观序列 $w_1t_{M_i}w_2$; 也就是说, 修改后的网络不是弱可诊断的。

另一方面, 如果 M 是可达的, 那么在 M 中, 变迁 t_f 可以触发, 之后, 只有变迁 t_a 可用, 因为 p_a 中有一个标记, 而 P 中没有标记。由于 t_a 的标签是 a, 这是一个新标签, 因此不存在具有相同观测但不包含 t_f 的序列。因此, 修改后的网络是弱可诊断的。 □

5.2.2　弱可预测性的可判定性

本小节将弱可预测性的研究重点从 NFA 转移到 LPN。首先, 在 LPN 框架内给出弱可预测性的定义, 如下所示。

定义 5.6 (LPN的弱可预测性)**.** 一个LPN系统 $G=(N,M_0,\Sigma,\ell)$ 对于故障变迁 T_f 是弱可预测的如果

$$(\exists K\in\mathbb{N})(\exists s\in\Psi(T_f))(\exists s'\in\overline{s}:T_f\notin s')(\forall\sigma'\in L(N,M_0))(\forall\sigma\in L(N,M_0)/\sigma')$$
$$[(\ell(\sigma')=\ell(s'))\wedge(T_f\notin\sigma')\wedge(|\sigma|\geq K)\Rightarrow(T_f\in\sigma)].$$

下面证明, 无界 LPN 的弱可预测性是不可判定的。

定理 5.5. 对于无界LPN, 验证弱可预测性的问题是不可判定的。

证明. 考虑两个没有不可观变迁的LPN G_1 和 G_2。现在将包含问题[140] 归约到弱可预测性验证问题。

从 G_1 和 G_2 构造一个LPN G, 如图 5.6 所示。具体地, 创建 5 个库所 p_0 到 p_4, 并引入了一个新的标签 a。

现在证明 $L(G_1)\subseteq L(G_2)$ 当且仅当 G 不是弱可预测性的。如果 $L(G_1)\subseteq$

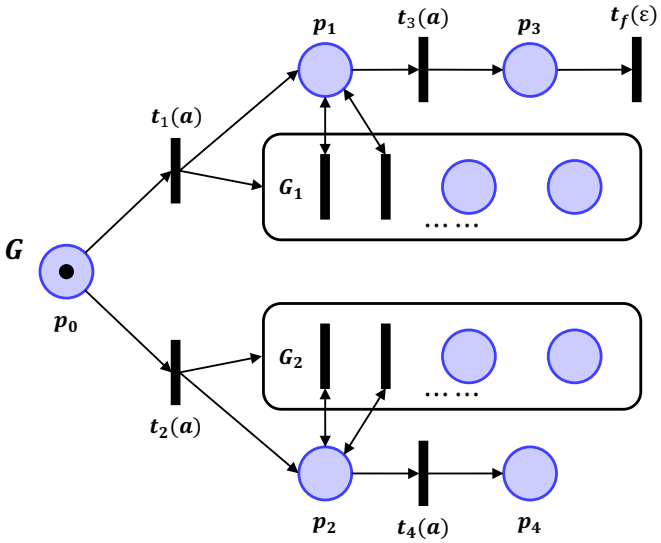

图 **5.6** 定理 **5.5** 归约证明的草图

$L(G_2)$, 那么任何形式为 $aw\varepsilon$ 的字符串, 其中 $w \in L(G_1)$, 对应于变迁序列 $t_1\sigma t_f$, 使得 $\sigma \in T_1$ 且 $\ell(\sigma) = w$, 可以通过 G_2 的对应部分进行模拟。因此, 在故障 t_f 发生之前, 没有观测可以导致故障被预测, 从而导致 G 不是弱可预测性的。

如果 $L(G_1) \not\subseteq L(G_2)$, 那么存在一个字符串 $w \in L(G_1)$ 使得 $w \notin L(G_2)$。设 $K = 1 \in \mathbb{N}$, 存在 $s = t_1\sigma t_3 t_f$ 和 $s' = t_1\sigma t_3$, 其中 $\ell(\sigma) = w$。对于任何 σ' 使得 $\ell(\sigma') = \ell(s') = awa$, 可以断定, 对于任何长度 $|\sigma| \geq 1$ 的后续序列, 故障 t_f 将肯定会发生。因此, G 是弱可预测性的。 □

5.3 本章小结

本章研究了验证 (模块化) NFA和LPN的弱可诊断性和弱可预测性的计算复杂度和可判定性。本章的结果显示, 对于NFA, 判定弱可诊断性和弱可预测性是PSPACE-complete; 对于模块化NFA, 这两个问题是EXPSPACE-complete; 而对于无界LPN, 弱可诊断性是 Ackermann-hard 且可判定性未知, 而弱可预测性是不可判定的。本文的后续两章将视角转向带有时间信息的离散事件系统: 恒定时间自动机与时间区间自动机, 并研究其故障诊断及控制器综合问题。

第6章　恒定时间自动机的协同可诊断性分析

本章针对CTA, 提出了一种验证其协同可诊断性的算法, 并详细分析了该算法的复杂度。此外, 讨论了在带有时间信息框架下的离散事件系统可诊断性与协同可诊断性的等价关系。本章沿用上一章的假设 5.1。

6.1　协同可诊断性的定义

在分散式架构中, 事件观测分布在 n 个局部诊断代理中。每个诊断代理拥有一组独立的可观事件集, 并独立运行且不相互通信。可观事件集的分布如下: $\Sigma_o = \bigcup_{j=1}^{n} \Sigma_o^j$, 其中 Σ_o^j 是代理 j 的可观事件集。定义 $\Sigma_{uo} = \Sigma \setminus \Sigma_o$ 和 $\Sigma_{uo}^j = \Sigma \setminus \Sigma_o^j$。根据德摩根定律, 有 $\Sigma_{uo} = \bigcap_{j=1}^{n} \Sigma_{uo}^j$。协同可诊断性和 T-协同可诊断性定义如下:

定义 6.1 ((或 T-) 协同可诊断性)**.** 对于投影 $P^j : (\Sigma \times \mathbb{R}_{\geq 0})^* \to (\Sigma_o^j \times \mathbb{R}_{\geq 0})^*$, 其中 $j = 1, 2, \ldots, n$, 和故障事件集 $\Sigma_f \subseteq \Sigma_{uo}$, 当且仅当满足以下条件, 称CTA G 是 (或 T-) **可协同诊断的**:

$$
\begin{aligned}
&(\exists K \in \mathbb{N})\ (\text{或 } (\exists T \in \mathbb{R}_{\geq 0}))(\forall s \in L(G), \ell_{\text{last}}(s) \in \Sigma_f) \\
&(\forall st \in L(G), |t| \geq K\ (\text{或 } \mathbf{T}_{\text{last}}(st) - \mathbf{T}_{\text{last}}(s) \geq T)) \\
\Rightarrow &(\exists j \in \{1, 2, \ldots, n\})(\forall w \in L(G), P^j(w) = P^j(st))(\exists e_f \in \Sigma_f, e_f \in \ell(w)).
\end{aligned}
$$

在上述定义中设 $n = 1$, 可以得出CTA的可诊断性定义。通过下面的示例来说明CTA的 (协同) 可诊断性含义。此外, 由于变迁时间的影响, CTA的 (协同) 可诊断性与其底层的FSA的 (协同) 可诊断性有所不同。

例 6.1. 考虑如图 6.1 所示的CTA G, 其中 $\Sigma = \{a, b, u, e_f\}$, $\Sigma_{uo} = \{u, e_f\}$, 且 $\Sigma_f = \{e_f\}$。注意, G 的底层FSA是不可诊断的, 因为对于观测 ab, 故障 e_f 的发生无法检测。然而, 令 $\mu(0, e_f, 4) = \tau = 0$, 借助时间信息, CTA G 变得可诊断。

设诊断代理1的可观事件集为 $\Sigma_o^1 = \{a\}$, 不可观事件集为 $\Sigma_{uo}^1 = \{b, u, e_f\}$；诊断代理2的可观事件集为 $\Sigma_o^2 = \{b\}$, 不可观事件集为 $\Sigma_{uo}^2 = \{a, u, e_f\}$。接下来, 考虑CTA G 及其底层FSA的协同可诊断性。考虑两条无限逻辑序列 $e_f abu^k$ 和 $uabu^k$, 其中 k 是任意大的。对于诊断代理1 (或代理2) , 这两条序列的投影相同, 分别为 a (或 b) , 且仅不同于前者包含故障 e_f, 而后者无故障。这表明CTA G 的底层逻辑FSA不可协同诊断。然而, 保持 $\tau = 0$ 并考虑变迁的时间后, 诊断代理1 (或代

理2) 在观测到 $(a,1)$ (或 $(b,4)$) 时可检测到故障, 因为非故障路径对应不同的观测, 即 $(a,2)$ (或 $(b,3)$) 。因此, 系统 G 实际上是可协同诊断的。

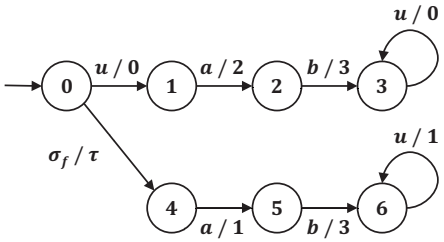

图 **6.1** 恒定时间自动机 G

简单来说, 如果至少有一个诊断代理能够在故障发生后的有限事件数内 (或在某个有限时间内) 检测到故障事件, 那么系统被认为是可协同诊断的 (或称 T-可协同诊断的) 。另一方面, 如果存在两个序列, 对于所有 n 个局部诊断代理, 它们的投影相同, 一个序列没有故障事件, 而另一个序列包含故障事件且在故障事件后是无限长的, 那么该系统就是不可协同诊断的。

6.2 协同可诊断性验证算法

本节提出了一种基于构建 **验证器** (Verifier) 以验证CTA的协同可诊断性的算法。基于验证器, 本节推导出了一个条件, 以降低可诊断性验证的复杂度。

6.2.1 正常子自动机和故障子自动机

用 $CoAc(G,Q_1)$ 表示 G 相对于状态子集 $Q_1 \subseteq Q$的共可达部分 (Coaccessible Part) 。这通过从 G 中删除所有没有通向 Q_1 中某个状态的路径的状态及其相关的输入和输出变迁来实现。

首先, 形式化定义CTA与FSA的乘积组合 (Product Composition) , 记作 $\boxtimes$, 以推导出原始系统的正常子自动机和故障子自动机, 这些将作为验证器的组件。为避免冗余, 本文仅考虑FSA的字母表是CTA字母表的子集的情况。

定义 6.2. 给定CTA $G_1 = (Q_1, \Sigma_1, Q_{i1}, \Delta_1, \mu_1)$ 和FSA $G_2 = (Q_2, \Sigma_2, Q_{i2}, \Delta_2)$, 其中 $\Sigma_2 \subseteq \Sigma_1$, 它们的**乘积组合**定义为CTA:

$$G_1 \boxtimes G_2 = Ac(Q_1 \times Q_2, \Sigma_2, Q_{i1} \times Q_{i2}, \Delta, \mu),$$

其中变迁关系 $\Delta \subseteq (Q_1 \times Q_2) \times \Sigma_2 \times (Q_1 \times Q_2)$ 和时间标签函数 $\mu : \Delta \to \mathbb{R}_{\geq 0}$ 被定义为: $((q_1, q_2), e, (q_1', q_2')) \in \Delta$ 和 $\mu((q_1, q_2), e, (q_1', q_2')) = \mu_1(q_1, e, q_1')$ 当且仅当 $(q_1, e, q_1') \in \Delta_1$ 和 $(q_2, e, q_2') \in \Delta_2$。

请注意, $G_1 \boxtimes G_2$ 的字母表被定义为较小的字母表 Σ_2 (即 G_1 和 G_2 字母表的交集) , 而不是较大的字母表 Σ_1 (即字母表的并集) 。这种方式与 [2] 中的经典乘积组合略有不同, 因为在这里仅关注被定义的变迁的标签。

算法 6.1 构造正常和故障子自动机

输入: 一个 CTA $G = (Q, \Sigma, Q_i, \Delta, \mu)$;
n 个局部诊断代理的事件集 $\Sigma = \Sigma_o^j \cup \Sigma_{uo}^j$, 其中 $j = 1, 2, \ldots, n$。
输出: 一组 n 个重新标记的正常部分 $G_N^j, j = 1, 2, \ldots, n$ 以及一个故障部分 G_F。

1: 按如下步骤构造重新标记的正常部分 G_N^j:
– 构建FSA $A_N = (\{N\}, \Sigma \setminus \Sigma_f, \{N\}, \{(N, e, N) \mid e \in \Sigma \setminus \Sigma_f\})$。
– 构造正常子自动机 $G_N = G \boxtimes A_N = (Q_N, \Sigma_N = \Sigma \setminus \Sigma_f = \Sigma_o^j \cup \Sigma_{N,uo}^j, Q_{i,N}, \Delta_N, \mu_N)$。
– 重新标记G_N为 $G_N^j = (Q_N, \Sigma_N^j = \Sigma_o^j \cup \Sigma_{N,uo}^{j,R}, Q_{i,N}, \Delta_N^j, \mu_N^j)$, 其中 $\Sigma_{N,uo}^{j,R} = \{e^j : e \in \Sigma_{N,uo}^j\}$ 以及

$$\begin{cases} \big((p, e, q) \in \Delta_N^j \Leftrightarrow (p, e, q) \in \Delta\big) \wedge \mu_N^j(p, e, q) = \mu_N(p, e, q) & \text{对于} e \in \Sigma_o^j; \\ \big((p, e^j, q) \in \Delta_N^j \Leftrightarrow (p, e, q) \in \Delta\big) \wedge \mu_N^j(p, e^j, q) = \mu_N(p, e, q) & \text{对于} e \in \Sigma_{N,uo}^j. \end{cases}$$

2: 按如下步骤构造故障部分 G_F:
– 构建FSA $A_l = (\{N, F\}, \Sigma, \{N\}, \Delta_{A_l})$, 其中 $\Delta_{A_l} = \{(N, e_f, F)\} \cup \{(N, e, N) \mid e \in \Sigma \setminus \Sigma_f\} \cup \{(F, e, F) \mid e \in \Sigma\}$。
– 构造 $G_l = G \boxtimes A_l = (Q_l, \Sigma, Q_{i,l}, \Delta_l, \mu_l)$。
– 构造故障部分 $G_F = CoAc(G_l, Q_l^F) = (Q_F, \Sigma, Q_{i,F}, \Delta_F, \mu_F)$, 其中 $Q_l^F = \{(q, q_l) \in Q_l : q_l = F\}$。

算法 6.1中, 步骤 1 构造了正常部分 G_N, 该部分建模了 G 的正常行为。简单来说, G_N 是通过从 G 中删除所有标记为故障事件 $e_f \in \Sigma_f$ 的变迁, 然后取结果自动机的可达部分来创建的。请注意, 在 G_N 中, 根据定义 6.2, 有 $\Sigma_{N,uo}^j = \Sigma_{uo}^j \setminus \Sigma_f$, 因为FSA A_N 的事件集为 $\Sigma \setminus \Sigma_f$。获得 G_N 后, 将 G_N 中 $\Sigma_{N,uo}^j$ 的不可观事件重新标记, 得到 G_N^j 中的 $\Sigma_{N,uo}^{j,R}$ $(j = 1, 2, \ldots, n)$, 以区分局部正常部分 G_N^j 中的不可观事件与故障部分 G_F 和其他局部正常部分 G_N^k $(k = 1, 2, \ldots, n$, 且 $k \neq j)$ 中的不可观事件。这意味着任何两个局部正常部分之间没有共享的不可观事件, 即 $\Sigma_{N,uo}^{j,R} \cap \Sigma_{N,uo}^{k,R} = \emptyset$ $(j, k \in \{1, 2, \ldots, n\}$ 且 $j \neq k)$ 。因此, 它们各自的不可观事件都是它们自己的私有事件, 即 $\Sigma_{N,uo}^{j,R} \subseteq \Sigma_N^j \setminus \Sigma_N^k$ $(j, k \in \{1, 2, \ldots, n\}$ 且 $j \neq k)$ 。

步骤 2 构造故障部分 G_F 以建模 G 的故障行为。简单来说, 将 G 的状态标

记为 F 如果在到达这些状态之前发生了故障事件, 然后将剩余的状态标记为 N。最后, 取结果自动机中标记为 F 的状态的共可达部分以获得 G_F。值得注意的是, 由于 A_l 的事件集为 Σ, 故障部分 G_F 与 G 共享字母表 Σ。此外, 由于 G_N^j 的所有不可观事件都已重新标记, 而故障事件仅存在于 G_F 中, 因此 $\Sigma_{N,uo}^{j,R} \cap \Sigma_{uo} = \emptyset$ 且 $\Sigma_N^j \setminus \Sigma = \Sigma_{N,uo}^{j,R}, j = 1, 2, \ldots, n$, 并且 $\Sigma \setminus \bigcup_{j=1}^{n} \Sigma_N^j = \Sigma_f$。由于 G_N 和 G_F 都是原始系统 G 的一部分, 因此 $L(G_N)$ 和 $L(G_F)$ 都是 $L(G)$ 的子集。

例 6.2. 再次考虑图 6.1 中的 CTA G, 其中 $\Sigma_o^1 = \{a\}$, $\Sigma_{uo}^1 = \{b, u, e_f\}$, $\Sigma_o^2 = \{b\}$, 和 $\Sigma_{uo}^2 = \{a, u, e_f\}$。按照算法 6.1, 可以构造出重新标记的正常部分 G_N^1, G_N^2, 和故障部分 G_F, 这些构造如图 6.4 所示, 其中 $\Sigma_N^1 = \Sigma_o^1 \cup \Sigma_{N,uo}^{1,R} = \{a\} \cup \{u^1, b^1\}$ 和 $\Sigma_N^2 = \Sigma_o^2 \cup \Sigma_{N,uo}^{2,R} = \{b\} \cup \{u^2, a^2\}$

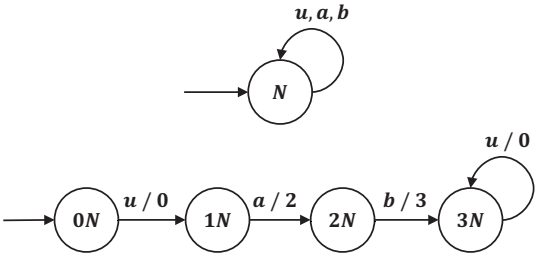

图 **6.2 FSA** A_N (上) 和正常部分 $G_N = G \boxtimes A_N$ (下)

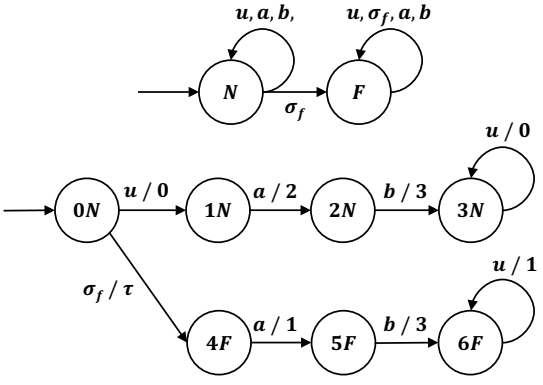

图 **6.3 FSA** A_l (上) 和 $G_l = G \boxtimes A_l$ (下)

6.2.2 验证器的构建

本小节将之前获得的重新标记的正常部分 G_N^j 和故障部分 G_F 组合起来构建验证器。值得注意的是, $\{G_F, G_N^1, G_N^2, \ldots, G_N^n\}$ 中任意两个组件之间的公有事件是

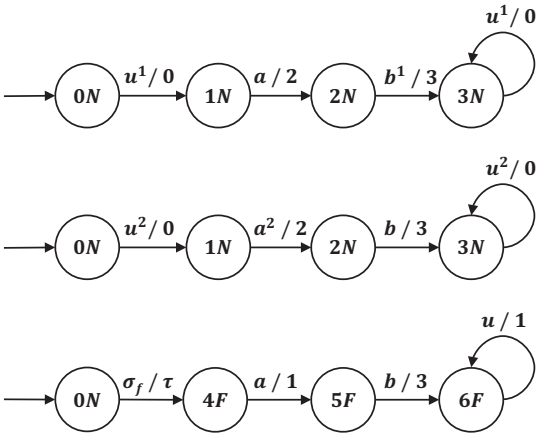

图 **6.4** 重新标记的正常部分 G_N^1, G_N^2 (上二) 和故障部分 G_F (下)

可观的；换句话说, 它们各自的不可观事件都是它们自己的私有事件。基于这一特性, 本节定义了两个 CTA 的时间并行组合 (Timed Parallel Composition) 。

定义 6.3. 给定两个 CTA $G_1 = (Q_1, \Sigma_1, Q_{i1}, \Delta_1, \mu_1)$ 和 $G_2 = (Q_2, \Sigma_2, Q_{i2}, \Delta_2, \mu_2)$, 且 $\Sigma_1 \cap \Sigma_2 = \Sigma_o^1 \cap \Sigma_o^2$, 它们的**时间并行组合**是定义在时间字母表上的 FSA, 其定义为

$$G_1 \parallel G_2 = Ac(Q_1 \times Q_2, \Sigma_{1\|2}, \Delta_{1\|2}, Q_{i1} \times Q_{i2}),$$

其中, $\Sigma_{1\|2} \subseteq (\Sigma_1 \cup \Sigma_2) \times \mathbb{R}_{\geq 0}$ 是时间标签的集合, $\Delta_{1\|2} \subseteq (Q_1 \times Q_2) \times \Sigma_{1\|2} \times (Q_1 \times Q_2)$ 是变迁的集合, 定义如下:

- 对于 $e \in \Sigma_o^1 \cap \Sigma_o^2$, $\big((p_1, p_2), (e, \tau), (q_1, q_2)\big) \in \Delta_{1\|2}$ 当且仅当在 G_1 中存在路径 π_1 且在 G_2 中存在路径 π_2 满足:

  $$\pi_1 = p_1 \xrightarrow{v_1} r_1 \xrightarrow{e} q_1,\ \pi_2 = p_2 \xrightarrow{v_2} r_2 \xrightarrow{e} q_2,$$

  其中 $\mu_1(p_1, v_1 e, q_1) = \mu_2(p_2, v_2 e, q_2) = \tau$, 且 $v_1 \in (\Sigma_{uo}^1)^*$, $v_2 \in (\Sigma_{uo}^2)^*$;

- 若 $e \in \Sigma_1 \setminus \Sigma_2$, $\big((p_1, p_2), (e, \mu_1(p_1, e, q_1)), (q_1, p_2)\big) \in \Delta_{1\|2}$ 当且仅当 $(p_1, e, q_1) \in \Delta_1$;

- 若 $e \in \Sigma_2 \setminus \Sigma_1$, $\big((p_1, p_2), (e, \mu_2(p_2, e, q_2)), (p_1, q_2)\big) \in \Delta_{1\|2}$ 当且仅当 $(p_2, e, q_2) \in \Delta_2$。

定义 6.3的第一项与文献 [76] 中方程 (6) 的直观含义相同。简单来说, 定义 6.3不仅考虑可观公有事件 $\Sigma_1 \cap \Sigma_2$ 和不可观私有事件 $(\Sigma_1 \setminus \Sigma_2) \cup (\Sigma_2 \setminus \Sigma_1)$ 的标签, 还考虑它们的持续时间。使用重新标记的正常部分和故障部分的时间并行组合, 从

而构建了验证器 $G_V = (Q_V, \Sigma_V, Q_{i,V}, \Delta_V, \mu_V) = (\|_{j=1}^{n} G_N^j) \parallel G_F$。

注意到 $\Sigma_N^j \setminus \Sigma = \Sigma_{N,uo}^{j,R}$ 和 $\Sigma \setminus \bigcup_{j=1}^{n} \Sigma_N^j = \Sigma_f$, 因此在构建 G_V 时, 不涉及任何组件的私有可观事件。因此, 定义 6.3 中省略了一些未使用的情况。具体而言, 在 G_V 中, 对于公有的可观事件, 仅构建两个部分中由相同事件和相同发生时间标签的变迁；对于私有不可观事件, 各组件则异步进行。

例 6.3. 使用图 6.4 中重新标记的正常部分 G_N^1、G_N^2 和故障部分 G_F, 在图 6.5 (或图 6.6) 中构建了验证器 $G_V = G_N^1 \parallel G_N^2 \parallel G_F$, 其中故障事件 e_f 的持续时间等于 0 或满足 $\tau + 1 \neq 2$ 的任意数值 (或等于 1) 。在FSA G_V 中, 每个事件都附有一个时间标签来记录其持续时间。例如, 在图 6.6 中, $(a, 2) \in \Sigma_V \subseteq (\Sigma \times \mathbb{R}_{\geq 0})$ 表示事件 a 的持续时间为 2。

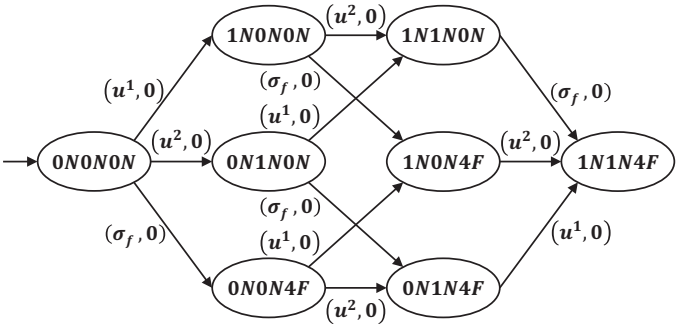

图 **6.5** 验证器 $G_V = G_N^1 \parallel G_N^2 \parallel G_F$ (当 $\mu_F(0N, e_f, 4F) = \tau = 0$)

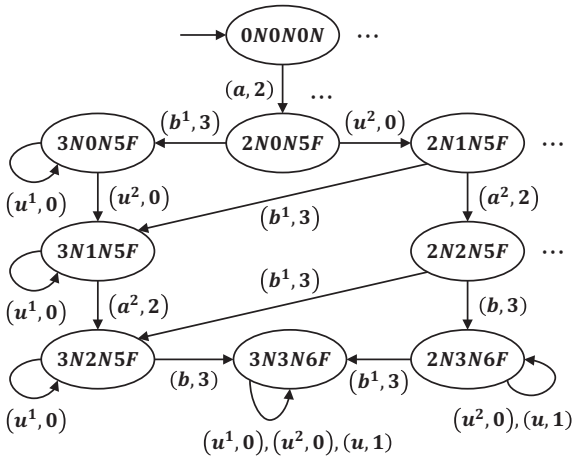

图 **6.6** 验证器 $G_V = G_N^1 \parallel G_N^2 \parallel G_F$ 的一部分 (当 $\mu_F(0N, e_f, 4F) = \tau = 1$)

注意, FSA G_V 的语言 $L(G_V)$ 是 Σ_V^* 的子集, 即 $L(G_V) \subseteq \Sigma_V^* \subseteq (\Sigma \times \mathbb{R}_{\geq 0})^*$, 而 G 的语言 $L(G)$ 是 $(\Sigma \times \mathbb{R}_{\geq 0})^*$ 的子集, 即 $L(G) \subseteq (\Sigma \times \mathbb{R}_{\geq 0})^*$。对于任何观测 $\sigma =$

$(e_1,\tau_1)(e_2,\tau_2)\cdots(e_k,\tau_k)\in P(L(G))$, 定义 $\sigma^{\text{elem}}=(e_1,\tau_1)(e_2,\tau_2-\tau_1)\cdots(e_k,\tau_k-\tau_{k-1})$ 来表示 σ 在 Σ_V^* 中的等价表达式。这里 $\tau_i-\tau_{i-1}$ $(i=2,3,\ldots,k)$ 表示可观事件 e_i 的持续时间。用 $P(L(G))^{\text{elem}}$ 表示 $P(L(G))$ 的等价表示形式, 即以事件的持续时间而非发生时间为准。以下命题展示了FSA G_V 与CTA G 之间的关系, 该结论可直接由G_V的构建过程及CTA定时字符串的定义自然导出。

命题 6.1. $P(L(G))^{\text{elem}}=P(L(G_V))$. □

6.2.3　协同可诊断性验证

用 $q_V=(q_N,q_F)\in Q_V$, 其中 $q_N=(q_N^1,q_N^2,\ldots,q_N^n)$, 分别表示来自 G_N^j 和 G_F 的状态, 并用 $q_F=(q,q_l)$ 表示来自 G 和 A_l 的状态。

定理 6.1. 当且仅当在验证器 G_V 中存在一个可达的故障状态循环, 并且该循环中至少包含一个来自 Σ 的事件标签时, CTA G 对于投影 P^j 和故障集 Σ_f 来说是不可协同诊断的。即存在一个循环 $(q_V^r,e_r,q_V^{r+1},\ldots,q_V^m,e_m,q_V^r)$, 其中 $q_V^i=(q_N^i,(q^i,q_l^i))\in Q_V, i\in\mathbb{N}_{rm}=\{r,r+1,\ldots,m\}$ 且 $m\geq r$, 满足以下条件:

$$(\exists i\in\mathbb{N}_{rm},\ell(e_i)\in\Sigma)\wedge(\forall i\in\mathbb{N}_{rm},q_l^i=F). \tag{6-1}$$

证明. "⇐": 假设在 G_V 中存在一个可达的循环 $(q_V^r,e_r,q_V^{r+1},\ldots,q_V^m,e_m,q_V^r)$, 并满足公式 (6-1) 中的条件。在这种情况下, 对于所有的 $K\in\mathbb{N}$, 存在一个时间标签序列 $s_Vt_V\in L(G_V)$, 使得 $e_f\in\ell(s_V)$, 其中 $e_f\in\Sigma_f$, 且 $t_V=(e_re_{r+1}\cdots e_m)^k$, 其中 $k\in\mathbb{N}$, 使得 $|t_V|>K$。进一步地定义以下映射:

$$P_N^j:\Big((\bigcup_{j=1}^n\Sigma_N^j\cup\Sigma)\times\mathbb{R}_{\geq0}\Big)^*\to(\Sigma_N^j\times\mathbb{R}_{\geq0})^*,$$
$$P_F:\Big((\bigcup_{j=1}^n\Sigma_N^j\cup\Sigma)\times\mathbb{R}_{\geq0}\Big)^*\to(\Sigma\times\mathbb{R}_{\geq0})^*.$$

鉴于 $\{\Sigma_{uo},\Sigma_{N,uo}^{1,R},\Sigma_{N,uo}^{2,R},\ldots,\Sigma_{N,uo}^{n,R}\}$ 中的任意两个事件集合没有公有事件, P_F (或 P_N^j) 可以被视为移除 $G_N^j(j=1,2,\ldots,n)$ 中的所有私有事件 (或移除其他满足 $k\neq j$ 的正常部分 G_N^k 及故障部分 G_F 中的所有私有事件) 的操作。由于 $G_V=(\|_{j=1}^n G_N^j)\parallel G_F$, 根据命题 6.1, 存在一个定时字符串 $st\in L(G_F)$, 使得 $s^{\text{elem}}=P_F(s_V)$, $t^{\text{elem}}=P_F(t_V)$, 且 $s^{\text{elem}}t^{\text{elem}}=P_F(s_Vt_V)$。由于 $t_V=(e_re_{r+1}\cdots e_m)^k$, 其中 $\ell(e_i)\in\Sigma$ 对于某些 $i=r,r+1,\ldots,m$, 因此, 满足 $t^{\text{elem}}=P_F(t_V)$ 的定时字符串 t 是任意长

的。因此, 故障事件 e_f 发生后, $st \in L(G_F) \subseteq L(G)$ 是任意长的。

同样, 根据命题 6.1, 存在一个定时字符串 $w_j \in L(G_N^j)$, 使得 $w_j^{\text{elem}} = P_N^j(s_V t_V)$ 对于 $j = 1, 2, \ldots, n$。由于 G_N^j 是通过重新标记 $\Sigma_{N,uo}^j$ 中的不可观事件从 G_N 得到的, 即 $\Sigma_N^j = \Sigma_o^j \cup \Sigma_{N,uo}^{j,R}$, 因此 w_j 中的所有不可观事件都属于 $\Sigma_{N,uo}^j$。这意味着存在一个定时字符串 $w \in L(G_N)$, 使得 $P^j(w) = P_F(w_j)$。由于 $\Sigma_N^j \cap \Sigma = \Sigma_o^j$ 而 $\Sigma_o^j \subseteq \Sigma_o$, 这表示 G_N^j 中的所有可观事件都是 G_N^j 和 G_F 之间的公有事件。根据 G_V 的构造, 在 $(\|_{j=1}^n G_N^j) \parallel G_F$ 中存在一个由可观事件 e 标记的变迁, 意味着在所有 G_N^j 和 G_F 中都有一个事件 e , 具有相同发生时间 (或者在 G_F 和某些 G_N^j 中, 其中 e 是它们相对于其他部分 G_N^k $(k \neq j)$ 的私有事件) 。结合 $\Sigma_{N,uo}^{j,R} \cap \Sigma_{uo} = \emptyset (j = 1, 2, \ldots, n)$, 有 $P_F(w_j) = P_N^j(st)$, 这意味着 $P^j(w) = P_N^j(st)$。由于定时字符串 $st \in L(G_F)$ 不包含 $\Sigma_{N,uo}^{j,R}$ 中的事件, 因此有 $P_N^j(st) = P^j(st)$, 从而 $P^j(w) = P^j(st)$。由于 $w \in L(G_N) \subseteq L(G)$, 即 w 中没有故障事件, 说明 G 不是可协同诊断的。

"⇒": 假设 G 对于 P^j 和 Σ_f 不满足协同可诊断性条件。那么对于任意的 $K \in \mathbb{N}$, 存在一个定时字符串 $st \in L(G_F)$, 其中 $\ell_{last}(s) \in \Sigma_f$ 且 $|t| > K$, 以及 $w \in L(G_N)$, 使得对所有 $j = 1, 2, \ldots, n$, 都有 $P^j(st) = P^j(w)$。由于定时字符串 st 在故障事件 e_f 发生后的长度是任意长的, 并且 G_F 具有有限数量的状态, 根据鸽巢原理, G_F 中必然存在一个循环。因为 st 在 $L(G_F)$ 中, 这个循环必须由故障状态组成。注意到 G_F 的事件集是 Σ, 因此循环中至少有一个事件属于 Σ。由于 $P^j(st) = P^j(w) = P_F(w_j)$, 其中 $w_j \in L(G_N^j)$ 是通过重新标记 w 中的不可观事件得到的, 因此根据 G_V 的构造, G_N^j 中的可观事件必然是 G_N^j 和 G_F 之间的公有事件, 其发生时间相同。注意到 $\Sigma_{N,uo}^{j,R} \cap \Sigma_{uo} = \emptyset, (j = 1, 2, \ldots, n)$, 因此 G_N^j 或 G_F 中的不可观事件必然是它们自己的私有事件。综上所述, 总是可以在 G_V 中构建一个满足公式 (6-1) 的循环。

□

例 6.4. 重新考虑图 6.5 中的 FSA G_V, 其中 $\mu(0, e_f, 4) = \mu_F(0N, e_f, 4F) = \tau = 0$, 以及图 6.6 中的 FSA G_V, 其中 $\tau = 1$。可以看出, 在图 6.5 中, 没有可达的循环满足公式 (6-1) 的条件。因此, 根据定理 6.1, 系统 G 是可协同诊断的。

在图 6.6 中, 存在两个包含 Σ 中事件的循环: $3N3N6F \xrightarrow{u/1} 3N3N6F$ 和 $2N3N6F \xrightarrow{u/1} 2N3N6F$。因此, CTA G 不是可协同诊断的。这一结论还可以通过检验 G 中的两个无限定时字符串来确认, 即 $(e_f, 1)(a, 2)(b, 5)(u, 6)(u, 7)\cdots$ 和 $(u, 0)(a, 2)(b, 5)(u, 5)(u, 5)\cdots$。这两个序列在诊断代理 1 (或 2) 中具有相同的投影 $(a, 2)$ (或 $(b, 5)$) , 但一个包含故障 e_f, 另一个则没有故障。

6.2.4　复杂度分析

在 [68] 中, 证明了判定 FSA 的协同可诊断性问题是 PSPACE-complete, 这意味着对于 FSA 及其他具有更复杂结构的模型, 几乎不可能存在多项式时间算法。因此, 本小节将专注于 CTA 的可诊断性验证的复杂度分析。本节首先展示了由于变迁时间的影响, 复杂度显著增加, 超出了逻辑情况下的多项式复杂度。随后, 本节提出了一个条件来将 CTA 的可诊断性验证复杂度降低到多项式时间。

算法 6.1 中所有自动机的最大状态数和变迁数如表 6.1 所示。由于其非确定性, CTA G 中每个状态的最大变迁数是 $|Q||\Sigma|$。算法 6.1 中自动机的构造依赖于逻辑操作, 因此它的复杂度和 G_N 和 G_F 的变迁数呈线性关系, 导致其多项式复杂度。

表 **6.1**　算法 **6.1** 中自动机的最大状态数及变迁数

	状态数	变迁数
G	$\|Q\|$	$\|Q\| \times \|Q\| \times \|\Sigma\|$
A_N	1	$\|\Sigma\| - \|\Sigma_f\|$
G_N	$\|Q\|$	$\|Q\| \times \|Q\| \times (\|\Sigma\| - \|\Sigma_f\|)$
A_l	2	$2\|\Sigma\|$
G_l	$2\|Q\|$	$2\|Q\| \times 2\|Q\| \times \|\Sigma\|$
G_N^j	$\|Q\|$	$\|Q\| \times \|Q\| \times (\|\Sigma\| - \|\Sigma_f\|)$
G_F	$2\|Q\|$	$2\|Q\| \times 2\|Q\| \times \|\Sigma\|$

根据 $G_V = (\|_{j=1}^{n} G_N^j) \parallel G_F$, 其状态数为 $2|Q|^{n+1} = 2|Q| \times \underbrace{|Q| \times |Q| \times \cdots \times |Q|}_{n\text{ 次}}$。然而, 即使仅用于可诊断性验证, $G_V = G_N \parallel G_F$ 的变迁数仍然可能会非常大。以下分析采用假设 5.2 简化复杂度分析[①]。

构造 $G_V = G_N \parallel G_F$ 的复杂度主要来自于定义 6.3 中 $\Delta_{1\|2}$ 的第一项, 因为其他项可以在常数时间内验证。为了检验第一项, 对于任意两对状态和一个共享的可观事件, 需要分别在 G_N 和 G_F 中找到所有由不可观事件组成并以该可观事件结尾的简单路径 (Simple Path)[②]。这通过 DFS 实现[126]。众所周知, 使用DFS查找两个节点之间所有简单路径的复杂度为 $\mathcal{O}\big((\mathbf{V}+\mathbf{E}) \times \mathbf{P}\big)$, 其中 $\mathbf{V}$ 为节点数, $\mathbf{E}$ 为边数, $\mathbf{P}$ 为简单路径的数量。在完全图中, $\mathbf{P} = (\mathbf{V}-2)!$。因此, 构造定义 6.3 中 $\Delta_{1\|2}$ 的第一项具有指数复杂度。直观来看, 在没有假设 5.2 的情况下, 复杂度会变得更高,

① 该假设仅适用于 Σ_{uo}, 不适用于局部不可观事件集 Σ_{uo}^j, 因为 Σ_{uo}^j 中的不可观事件数量可能很大。

② 简单路径是指不经过重复节点的路径。由于假设 5.2 的存在, 仅需考虑简单路径。

因为两个节点之间可能存在无限多条不可观路径。

因此, 在最坏情况下, 上述每条简单路径可能具有不同的持续时间, 从而导致时间字母表 Σ_V 的基数是指数级的。因此 G_V 中的变迁数量呈指数增长。应用定理 6.1 来识别 G_V 中的非平凡强连通分量 (Strongly Connected Components, SCC) ①, 然后在这些 SCC 中检验公式 (6-1) 的条件。这个过程的复杂度与 G_V 的变迁数成线性[141], 因此最终需要指数时间。在文献 [76] 的定理 3.7 中, 即使对于确定性的最大加自动机 (Max-Plus Automata, MPA) , T-可诊断性验证问题也是 coNP-hard。注意, 确定性意味着非歧义性 (Unambiguity) 。结合文献 [73] 中的命题 1, 该命题建立了非歧义最大加自动机 (Unambiguous Max-Plus Automata, UMPA) 的可诊断性与 T-可诊断性的等价性, 本文将 [73] 中关于 UMPA 可诊断性验证复杂度的声明从多项式时间更正为 coNP-hard。

回想一下, 在 FSA 的框架下, 不可观事件的发生不会影响观测。然而, 识别变迁持续时间的必要性是导致 CTA 复杂度增加的原因。尽管如此, 这种在实际中难以处理的复杂度引发了一个问题: 是否存在结构上更简单的 CTA, 使得该问题变得可处理。本节推导出以下条件, 以降低 CTA 可诊断性验证的复杂度。

定理 6.2. 给定一个多项式函数 $\mathcal{P}:\mathbb{N}\to\mathbb{N}$, 如果一类 CTA $G=(Q,\Sigma,Q_i,\Delta,\mu)$ 满足假设 5.2, 并且满足以下条件

$$\max_{\forall q,q'\in Q}\sum_{w\in\Sigma_{uo}^*}|q\overset{w}{\rightsquigarrow}q'|\leq\mathcal{P}(|Q|), \tag{6-2}$$

则使用 $G_V=G_N\parallel G_F$ 判定 G 的可诊断性的复杂度为 $\mathcal{O}(|Q|^4\mathcal{P}(|Q|))$。

证明. 结合假设 5.2 和公式 (6-2) 可知, G 中任意两个状态之间的不可观简单路径的数量的上界是多项式函数 $\mathcal{P}$。利用上述的 DFS 查找简单路径的公式 $\mathcal{O}((\mathbf{V}+\mathbf{E})\times\mathbf{P})$, 可以推导出构造 $G_V=G_N\parallel G_F$ 的复杂度为 $\mathcal{O}(\binom{|Q|}{2}\times(|Q|+|Q|^2|\Sigma|)\times\mathcal{P}(|Q|)+\binom{2|Q|}{2}\times(2|Q|+4|Q|^2|\Sigma|)\times\mathcal{P}(|Q|))=\mathcal{O}(|Q|^4\mathcal{P}(|Q|))$。注意, 公式 (6-2) 还意味着 G_V 中任意两个状态之间的变迁数为 $\mathcal{P}(|Q|)$。因此, G_V 中的最大变迁数为 $\binom{2|Q|^2}{2}\mathcal{P}(|Q|)$。由于定理 6.1 的验证与 G_V 中变迁数成线性关系, 证毕。 □

值得注意的是, 如果假设 5.2、$\Sigma_{uo}=\Sigma_f=\{e_f\}$ 和非歧义性②同时成立, 那么 $\mathcal{P}(|Q|)$ 被 $|Q|^3|\Sigma|$ 上界约束。这是因为在这种情况下, 任意两个状态之间最多存在

① 如果一个 SCC 是单个没有自环的节点, 则为平凡的 (trivial) ；否则为非平凡的 (nontrivial) 。

② 如果对于 G 中的任何状态 q 和 Σ^* 中的任何字符串 w, 最多只有一条以 w 为标记的路径从初始状态通向 q, 则称 CTA 是非歧义的, 该性质可以在多项式时间内检验[142] 。

$|Q|-1$ 条路径。因此, 文献 [73] 中关于多项式复杂度的声明在此条件下成立。此外, 将非歧义性替换为确定性时, $\mathcal{P}(|Q|)$ 的上界将降低为 1。此外, 文献 [74-75] 中的观测器或文献 [77] 中的诊断器无法使用定理 6.2 中的条件, 因为它们在最坏情况下可能具有指数级的状态数 (不仅仅是变迁数) 。

6.3　协同可诊断性和可诊断性的关系

本节做出一个额外假设, 并证明在 CTA 框架中, 协同可诊断性可以与可诊断性等价, 而在 FSA 框架中无法得出相同的结论。

假设 6.1. 系统 G 中每个变迁的持续时间是严格正的。

定理 6.3. 一个 CTA G 对于投影 $P^j:(\Sigma\times\mathbb{R}_{\geq 0})^*\to(\Sigma_o^j\times\mathbb{R}_{\geq 0})^*$ $(j=1,2,\ldots,n)$ 和故障事件集 Σ_f 是可协同诊断的, 当且仅当它对于投影 $P:(\Sigma\times\mathbb{R}_{\geq 0})^*\to(\Sigma_o\times\mathbb{R}_{\geq 0})^*$ 和故障事件集 Σ_f 是可诊断的, 其中 $\Sigma_o=\bigcup_{j=1}^n\Sigma_o^j$。

证明. "⇒": 假设 G 对于投影 $P:(\Sigma\times\mathbb{R}_{\geq 0})^*\to(\Sigma_o\times\mathbb{R}_{\geq 0})^*$ (其中 $\Sigma_o=\bigcup_{j=1}^n\Sigma_o^j$) 不是可诊断的。这意味着存在两个定时字符串 $w_N,w_F\in L(G)$, 使得 $P(w_N)=P(w_F)$, 其中 $e_f\in\ell(w_F)$ 且 $e_f\in\Sigma_f$。由于 $\Sigma_o^j\subseteq\Sigma_o$, 对于所有局部诊断代理, 有 $P^j(w_N)=P^j(w_F)$。因此, G 对于投影 P^j 不是可协同诊断的。

"⇐": 假设 G 对于投影 $P:(\Sigma\times\mathbb{R}_{\geq 0})^*\to(\Sigma_o\times\mathbb{R}_{\geq 0})^*$ (其中 $\Sigma_o=\bigcup_{j=1}^n\Sigma_o^j$) 是可诊断的。这意味着存在一个自然数 $K\in\mathbb{N}$, 使得对于 $L(G)$ 中的任何两个定时字符串 w_N 和 st, 如果 $e_f\notin\ell(w_N)$、$\ell_{last}(s)\in\Sigma_f$, 且 $|t|\geq K$, 则 $P(w_N)\neq P(st)$。假设 $P(w_N)=(e_1,\tau_1)(e_2,\tau_2)\cdots(e_m,\tau_m)$ 且 $P(st)=(e_1',\tau_1')(e_2',\tau_2')\cdots(e_l',\tau_l')$。由于假设 6.1, 有 $\tau_1<\tau_2\cdots<\tau_m$ 和 $\tau_1'<\tau_2'\cdots<\tau_l'$, 即两条定时字符串中的事件发生时间点是严格递增的。根据 $P(w_N)\neq P(st)$ 的情况可以考虑两种情况: 要么这两个投影中的事件逻辑序列不同, 要么这两个投影中的事件逻辑序列完全相同, 但某些对应事件的发生时间点不同。对于第一种情况, 如果存在 $j\in\{1,2,\ldots,n\}$, 使得 $P^j(w_N)$ 中的事件逻辑序列与 $P^j(st)$ 中的事件逻辑序列不同, 则证明完成。对于第二种情况, 必然存在 $j\in\{1,2,\ldots,n\}$, 使得 $P^j(w_N)$ 中的事件逻辑序列与 $P^j(st)$ 中的事件逻辑序列相匹配, 但 $P^j(w_N)$ 和 $P^j(st)$ 中某些相同事件的持续时间不同, 因为它们的发生顺序不同。在这种情况下, 必然存在 $(e_k,\tau_k)\neq(e_k,\tau_k')$, 其中 $(e_k,\tau_k)\in P(w_N)$ 且 $(e_k,\tau_k')\in P(st)$。因此, 存在 $j\in\{1,2,\ldots,n\}$, 使得 $e_k\in\Sigma_o^j$ (由于 $\Sigma_o^j\subseteq\Sigma_o$) , 这意味着 $P^j(w_N)\neq P^j(st)$。因此, 即使 $P^j(w_N)$ 和 $P^j(st)$ 中的事件

逻辑序列保持不变, 特定的局部诊断代理仍然能够区分对应的相同事件的不同持续时间。因此, G 对于投影 $P^j : (\Sigma \times \mathbb{R}_{\geq 0})^* \to (\Sigma_o^j \times \mathbb{R}_{\geq 0})^*$ 是可协同诊断的。 □

值得注意的是, 在 FSA 框架中, 上述证明中的“⇒”部分可以推导出来, 而“⇐”部分则不成立。然而, “⇐”部分在 CTA 中是独特的, 因为事件的持续时间在确定事件发生的顺序中起着重要作用。

命题 6.2. CTA G 当且仅当它是可协同诊断时, 它也是 T-可协同诊断的。

证明. “⇐”: 假设 CTA G 是可协同诊断的, 这意味着至少有一个诊断代理 $j \in \{1, 2, \ldots, n\}$ 能够在故障发生后的 $K \in \mathbb{N}$ 个事件内检测到故障。对于每一条长度为 K 的路径, 这些路径从一个带有故障标签的输入变迁的状态出发, 这可以通过对路径上每个变迁的持续时间求和来计算这条路径的总持续时间。由于每段持续时间属于 $\mathbb{R}_{\geq 0}$, 因此路径的总持续时间是有限的。设 T 为这些路径持续时间的最大值。因此, 故障发生后, 诊断代理 j 必然在最多 T 时间单位内检测到故障, 这意味着 G 是 T-可协同诊断的。

“⇒”: 需要证明, 如果 G 不是可协同诊断的, 则它不能是 T-可协同诊断的。假设 G 不是可协同诊断的, 这意味着存在两个定时字符串 $s'_j \in L(G) \cap (\Sigma \setminus \Sigma_f)^*$ 和 $s_j e_f t_j \in L(G)$, 使得对于任意 $j \in \{1, 2, \ldots, n\}$, 都有 $P^j(s'_j) = P^j(s_j e_f t_j)$, 其中 $|t_j| \geq K \in \mathbb{N}$ 且 $e_f \in \Sigma_f$。若 K 趋近于正无穷, 则 $|s_j e_f t_j|$ 也趋近于正无穷。此外, 根据假设 6.1, 该假设要求每个变迁的持续时间都是正数, $\mathbf{T}_{\text{last}}(s_j e_f t_j) - \mathbf{T}_{\text{last}}(s_j e_f)$ 趋近于正无穷, 这意味着它不属于 $\mathbb{R}_{\geq 0}$。因此, G 不可能是 T-可协同诊断的。 □

注意, 通过取 $n = 1$, 可以得到 T-可诊断性和可诊断性之间的等价性。因此, 本节在 CTA 框架中推导出了四个概念: 可诊断性、协同可诊断性、T-可诊断性和 T-协同可诊断性之间的等价性。

6.4 本章小结

本章形式化定义了两个 CTA 的时间并行组合。基于此操作, 提出了一种验证 CTA 协同可诊断性的算法。最后, 本章证明了 CTA 框架中可诊断性和协同可诊断性之间的等价性, 这与 FSA 框架中的情况不同。下一章将拓展本章故障诊断的工作到更复杂的时间区间自动机模型上, 并研究相关控制器综合问题。

第7章　时间区间自动机的主动诊断

7.1　时间区间自动机的可诊断性验证

本节首先为给定的 TIA $G = (Q, \Sigma, Q_i, \Delta, \mu)$ 形式化定义了可诊断性的概念。此外, 本节为 G 构造了一个验证器, 用于验证其可诊断性。本节沿用章节 6 的两个标准假设 (假设 5.1 和 5.2) , 以确保原系统始终生成可观测的行为。

例 7.1. 考虑图 7.1 中的 TIA, 其中状态变迁表示了转移关系和特定的转移权重。例如, 状态变迁 $0 \xrightarrow{a/[1,2]} 1$ 指定了 $\mu(0, a, 1) = [1, 2]$。考虑 TIA G 中的一个运行：

$$\rho = 0 \xrightarrow{a/1.5} 1 \xrightarrow{e_f/1} 2 \xrightarrow{b/2} 4 \xrightarrow{c/1.5} 2,$$

则 ρ 的定时字符串为 $\mathcal{TW}(\rho) = (a, 1.5)(e_f, 2.5)(b, 4.5)(c, 6)$。其观测为 $P(\mathcal{TW}(\rho)) = (a, 1.5)(b, 4.5)(c, 6)$。进一步地, 有 $\ell(P(\mathcal{TW}(\rho))) = abc$。

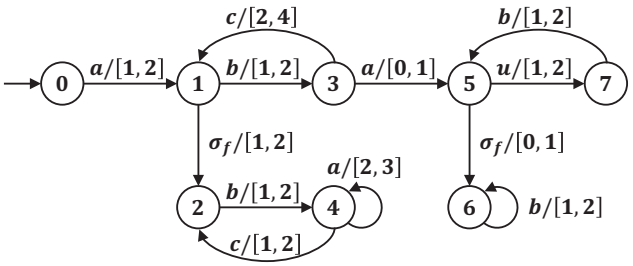

图 **7.1 TIA** G**,** 其中 $\Sigma_o = \{a, b, c\}$ 且 $\Sigma_{uo} = \{u, e_f\}$

7.1.1　可诊断性的定义

定义 7.1. 一个 TIA G 相对于投影 $P : (\Sigma \times \mathbb{R}_{\geq 0})^* \to (\Sigma_o \times \mathbb{R}_{\geq 0})^*$ 和故障事件集 $\Sigma_f \subseteq \Sigma_{uo}$ 是**可诊断的**, 当且仅当满足以下条件：

$$(\exists K \in \mathbb{N})(\forall s \in L(G), \ell_{last}(s) \in \Sigma_f)(\forall w = st \in L(G))$$
$$[|t| \geq K \Rightarrow (\forall w' \in L(G), P(w') = P(w))(\exists e_f \in \Sigma_f, e_f \in \ell(w'))]$$

简单来说, 对于一个 TIA, 如果存在一个自然数 $K \in \mathbb{N}$, 使得每当故障事件 $e_f \in \Sigma_f$ 发生时, 它可以在接下来发生的 K 个事件内被检测到, 那么该自动机就是可诊断的。这意味着系统能够在故障发生后不久可靠地识别故障, 确保及时检测。

7.1.2 验证器的构建

为了构造一个 TIA 的验证器, 先为两个TIA定义如下的时间并行组合操作。

定义 7.2. 给定两个 TIA $G_1 = (Q_1, \Sigma_1, Q_{i1}, \Delta_1, \mu_1)$ 和 $G_2 = (Q_2, \Sigma_2, Q_{i2}, \Delta_2, \mu_2)$, G_1 和 G_2 的**时间并行组合**定义为一个TIA

$$G_1 \parallel G_2 = Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, Q_{i1} \times Q_{i2}, \Delta_{1\|2}, \mu_{1\|2}),$$

其中 $\Delta_{1\|2} \subseteq (Q_1 \times Q_2) \times (\Sigma_1 \cup \Sigma_2) \times (Q_1 \times Q_2)$ 和 $\mu_{1\|2} : \Delta_{1\|2} \to 2^{\mathbb{R}_{\geq 0}} \setminus \{\emptyset\}$ 定义为：

- 对于 $\sigma \in \Sigma_1 \cap \Sigma_2$, $((p_1, p_2), \sigma, (q_1, q_2)) \in \Delta_{1\|2}$ 且 $\mu_{1\|2}((p_1, p_2), \sigma, (q_1, q_2)) = \mu_1(p_1, \sigma, q_1) \cap \mu_2(p_2, \sigma, q_2)$ 当且仅当 $\mu_1(p_1, \sigma, q_1) \cap \mu_2(p_2, \sigma, q_2) \neq \emptyset$.
- 对于 $\sigma \in \Sigma_1 \backslash (\Sigma_1 \cap \Sigma_2)$, $((p_1, p_2), \sigma, (q_1, p_2)) \in \Delta_{1\|2}$ 且 $\mu_{1\|2}((p_1, p_2), \sigma, (q_1, p_2)) = \mu_1(p_1, \sigma, q_1)$ 当且仅当 $(p_1, \sigma, q_1) \in \Delta_1$.
- 对于 $\sigma \in \Sigma_2 \backslash (\Sigma_1 \cap \Sigma_2)$, $((p_1, p_2), \sigma, (p_1, q_2)) \in \Delta_{1\|2}$ 且 $\mu_{1\|2}((p_1, p_2), \sigma, (p_1, q_2)) = \mu_2(p_2, \sigma, q_2)$ 当且仅当 $(p_2, \sigma, q_2) \in \Delta_2$.

简单来说, 两个 TIA 之间的时间并行组合同时考虑了公有事件 $\Sigma_1 \cap \Sigma_2$ 和私有事件 $(\Sigma_1 \cup \Sigma_2) \setminus (\Sigma_1 \cap \Sigma_2)$ 的标签和时间信息。公有事件仅在两个TIA都能同时执行时才会被执行, 这表示两个组件中的对应经过时间区间相交。然而, 私有事件可以在任何可能的情况下执行。注意这与上一章中两个 CTA 之间的时间并行组合略有不同。

基于上述的定时并行组合, 算法 7.1 描述了用于可诊断性验证的验证器的构造过程。该算法的基本直觉是区分正常观测和故障观测。算法 7.1 的第 1 步构造标签自动机 G_l, 它在语言上与 G 等价。G_l 的状态形式为 (q, q_l), 其中 $q_l \in \{N, F\}$。附加在状态 q 上的标签取决于从初始状态到达 q 的字符串中是否包含故障事件 $e_f \in \Sigma_f$ (标记为 N 表示缺失, 标记为 F 表示包含) 。为简化起见, 每个属于 $Q \times \{N, F\}$ 的状态使用简写表示, 例如 $1N$ 代表 $(1, N)$。接下来, 从 G_l 推出观测自动机 G_o。在 G_o 中, 仅保留初始状态以及在 G_l 中有被可观事件标记的输入变迁的状态。此外, 对于任何字符串 $\nu\sigma \in \Sigma^*$, 其中 $\nu \in \Sigma_{uo}^*$ 且 $\sigma \in \Sigma_o$, 将 ν 的权重添加到 σ 的权重上。值得注意的是, 在状态 p 和状态 q 之间可能存在多个由逻辑字符串 $\nu_1\sigma, \nu_2\sigma, \ldots$ 标记的路径。在这种情况下, 将变迁 $(p, \sigma, q) \in \Delta_o$ 的权重设置为它们权重的并集。最

算法 7.1 验证器的构造

输入： TIA $G = (Q, \Sigma, Q_i, \Delta, \mu)$。
输出： 验证器 $G_v = (Q_v, \Sigma_o, Q_v^i, \Delta_v, \mu_v)$。

1: 构造标签自动机 G_l, 步骤如下：
– 设 $A_l = (\{N, F\}, \Sigma_f, \{N\}, \{N, F\} \times \Sigma_f \times \{F\}, \mu_l^R : \{N, F\} \times \Sigma_f \times \{F\} \to \{\mathbb{R}_{\geq 0}\})$。
– 计算标签自动机 $G_l = G \parallel A_l = (Q_l, \Sigma, Q_l^i, \Delta_l, \mu_l)$。
2: 从 G_l 构造观测自动机 $G_o = (Q_o, \Sigma_o, Q_o^i, \Delta_o, \mu_o)$, 步骤如下：
– $Q_o^i = Q_l^i$;
– $Q_o = Q_o^i \cup Q_o'$ 是初始状态与具有可观输入变迁的状态集合 Q_o' 的并集;
– 所有变迁权重初始时设为空集, 然后 $\Delta_o \subseteq Q_o \times \Sigma_o \times Q_o$ 和 $\mu_o : \Delta_o \to 2^{\mathbb{R}_{\geq 0}}$ 定义如下：对于任何 $\sigma \in \Sigma_o$ 和 $\nu \in \Sigma_{uo}^*$, 如果 $(p, \nu\sigma, q) \in \Delta_l$, 则 $(p, \sigma, q) \in \Delta_o$ 且 $\mu_o(p, \sigma, q) = \mu_o(p, \sigma, q) \cup \mu_l(p, \nu\sigma, q)$。
3: 构造验证器 $G_v = G_o \parallel G_o$。

后, 通过对 G_o 与自身的时间并行组合来构造验证器 G_v, 这意味着仅应用定义 7.2 中的变迁关系的第一个项。

例 7.2. 再次考虑图 7.1 中的TIA G, 其中 $\Sigma_f = \{e_f\} \subseteq \Sigma_{uo}$。根据算法 7.1 的前两步, 基于图 7.2 构造的观测自动机 G_o 如图 7.3 所示。根据定义 7.2, 基于 G_o 构造的验证器 G_v 如图 7.4 所示。

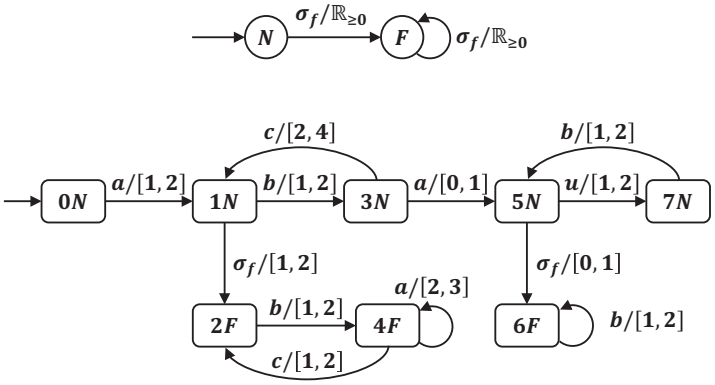

图 **7.2** 自动机 A_l (上) 和标签自动机 G_l (下)

7.1.3 可诊断性验证

定理 7.1. TIA G 在投影 P 和故障事件 Σ_f 下**不可诊断**, 当且仅当 G_v 中存在一个循环, 该循环仅由状态 $((q, l), (q', l'))$ 构成, 其中 $l \neq l'$。

证明. 由于在自动机 A_l 中没有相对于 G 的私有事件, 并且 $G_l = G \parallel A_l$ 没有限制 G 的逻辑或时间行为, 因此有 $L(G_l) = L(G)$。算法 7.1 的第 2 步在 G_o 中仅保留 G_l 的可观行为, 因此有 $L(G_o) = P(L(G_l))$。因此, $L(G_o) = P(L(G))$。

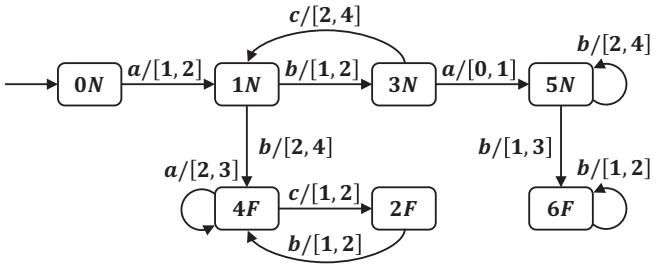

图 **7.3** 观测自动机 G_o

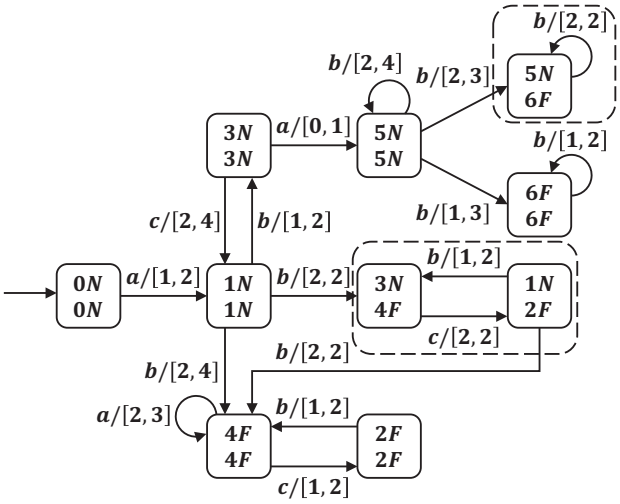

图 **7.4** 验证器 G_v **(**省略状态 $(4F,3N)$、 $(2F,1N)$ 和 $(6F,5N)$**)**

“⇐”: 假设在 G_v 中存在一个可达的循环

$$((q_1,l_1),(q_1',l_1')) \xrightarrow{\sigma_1} ((q_2,l_2),(q_2',l_2')) \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_{k-1}} ((q_k,l_k),(q_k',l_k')) \xrightarrow{\sigma_k} ((q_1,l_1),(q_1',l_1'))$$

其中 $l_1 \neq l_1', l_2 \neq l_2', \ldots, l_k \neq l_k'$ 且 $\sigma_1, \sigma_2, \ldots, \sigma_k \in \Sigma_o$。根据 G_v 的构造, 有 $l_i = N$ 且 $l_i' = F$ (或 $l_i = F$ 且 $l_i' = N$) , $i = 1, 2, \ldots, k$。假设这个循环从初始状态 $((q_0,N),(q_0,N))$ 通过时间字符串 w 可达。则必然存在两条运行

$$\rho_N = (q_0,N) \xrightarrow{w/\iota} (q_1,N) \xrightarrow{\sigma_1/\tau_1} (q_2,N) \xrightarrow{\sigma_2/\tau_2} \cdots \xrightarrow{\sigma_{k-1}/\tau_{k-1}} (q_k,N) \xrightarrow{\sigma_k/\tau_k} (q_1,N),$$
$$\rho_F = (q_0,N) \xrightarrow{w/\iota} (q_1',F) \xrightarrow{\sigma_1/\tau_1} (q_2',F) \xrightarrow{\sigma_2/\tau_2} \cdots \xrightarrow{\sigma_{k-1}/\tau_{k-1}} (q_k',F) \xrightarrow{\sigma_k/\tau_k} (q_1',F)$$

在 G_o 中, 且 $\mathcal{TW}(\rho_N) = \mathcal{TW}(\rho_F)$。注意标签 F 的出现是由于故障事件 $e_f \in \Sigma_f$ 的发生。结合 $P(L(G)) = L(G_o)$, 总是可以在 G 中找到两个任意长的时间字符串 (通过将循环重复任意多次) , 它们具有相同的投影。其中一个包含故障事件 e_f, 而另一个不包含。因此, TIA G 是不可诊断的。

“⇒”: 假设 TIA G 是不可诊断的。因此, 对于任何 $K \in \mathbb{N}$, 存在时间字符串

$s_F t_F, s_N \in L(G)$, 其中 $e_f \in \ell(s_F)$, $|t_F| > K$, 且 $e_f \notin \ell(s_N)$, 满足 $P(s_N) = P(s_F t_F)$。注意, $P(L(G)) = L(G_o)$ 成立, 并且在 $G_v = G_o \parallel G_o$ 中, 将通过相同观测到达的任意两个状态组合在一起。因此, 在 G_v 中必然存在一个循环, 其由左侧和右侧组件中具有不同标签的状态形成, 其中标签 N 来自 s_N, 标签 F 来自 $s_F t_F$。 □

例 7.3. 考虑图 7.4 中所示的验证器 G_v, 其中循环 (符合定理 7.1 的条件, 即导致不可诊断性的循环) 已用虚线框出。因此, 根据定理 7.1, G 是不可诊断的。

7.2　时间区间自动机的主动诊断

本节讨论**主动诊断**问题, 即为一个不可诊断的系统设计控制策略, 使其闭环系统变得可诊断。已知不可诊断性是由两个任意长的字符串具有相同的观测结果引起的, 其中一个包含故障, 而另一个不包含故障, 见如下示例。

例 7.4. 再次考虑图 7.1 中的 TIA G。可以很容易地验证出 G 的底层逻辑自动机是不可诊断的。更具体地说, 存在两个无限逻辑字符串 $a(bc)^k$ 和 $ae_f(bc)^k$, 其中 $k \in \mathbb{N}$ 任意大。值得注意的是, 这两个字符串中一个包含故障事件 e_f, 而另一个不包含。因此, G 对应的逻辑自动机是不可诊断的, 因为这两个任意长的逻辑字符串具有相同的观测结果 $a(bc)^k$。然而, 当考虑时间信息时, TIA 的可诊断性可能与其对应的逻辑自动机不同。如果现在将 G 中的变迁 $1 \xrightarrow{b/[1,2]} 3$ 替换为 $1 \xrightarrow{b/(4,5]} 3$, 根据算法 7.1, 可以推导出修改后的 TIA 的验证器仅包含变迁 $(0N, 0N) \xrightarrow{a/[1,2]} (1N, 1N)$。因此, 根据定理 7.1, 修改后的 TIA 是可诊断的。

上述现象体现了时间信息在 TIA 中的作用, 从而得出了本节的发现。在FSA中, 监督器通常会禁用某些可控事件, 以避免故障的发生。然而, 在 TIA 的时间字符串框架下, 可以有意限制某些变迁的发生时间点。这种有意的限制使得两个时间字符串在观测者看来有所不同, 从而暴露出故障的发生。因此, 时间信息提供了新的可诊断性使能视角。

在经典的监督控制中, 可控事件是监督器可以阻止其发生的事件, 而不可控事件则不能被禁用。然而, 在本章中, 可控事件可以被禁用或限制其权重, 但这两种操作都不适用于不可控事件。因此定义新的控制模式如下：

$$\Gamma = \{\gamma \subseteq Q \times \Sigma \times Q \times 2^{\mathbb{R}_{\geq 0}} \setminus \{\emptyset\} \mid Q \times \Sigma_{uc} \times Q \times 2^{\mathbb{R}_{\geq 0}} \setminus \{\emptyset\} \subseteq \gamma\},$$

其中 $(q, \sigma, q', I) \in \gamma$ 表示将 $(q, \sigma, q') \in \Delta$ 的权重 $\mu(q, \sigma, q')$ 限制为 $I \subseteq \mu(q, \sigma, q')$。

虽然通过监督器禁用一个变迁可以被视为将其权重限制为空集, 但需要注意的是, 空集从不作为变迁权重使用。换句话说, 与逻辑系统的控制一样, 禁用操作意味着从集合 Γ 中移除被禁用的变迁。与文献中具有部分观测的监督控制相同, 假设所有可控事件都可观。

假设 7.1. $\Sigma_c \subseteq \Sigma_o$ (即 $\Sigma_{uo} \subseteq \Sigma_{uc}$)。

现在形式化定义以下概念, 以便后续设计控制策略。为了方便起见, 将满足定理 7.1 条件的循环称为**模糊循环**。

定义 7.3. 给定一个 TIA G, 通过算法 7.1 得到 G 的验证器 G_v。在 G_v 中的模糊循环如果仅包含不可控事件, 则被称为**不可控模糊循环**；否则, 被称为**可控模糊循环**。

定义 7.4. 给定一个 TIA G, 其对应的验证器为 $G_v = (Q_v, \Sigma_v, Q_v^i, \Delta_v, \mu_v)$。如果存在一个从 q_v 到 $q_v' \in Q_v$ 的运行, 满足以下两个条件：1) 该运行仅包含不可控事件, 2) q_v' 属于一个不可控模糊循环, 则状态对 q_v 被称为**不可诊断状态对**。否则, 它被称为**可诊断状态对**。

在线算法 7.3 中涉及使用当前状态估计进行主动诊断。直观地说, 如果系统的当前状态估计包含了不可诊断状态对中的两个状态, 则系统不可避免地会违反诊断性。用 Q_{ndiag} 表示 G_v 中的不可诊断状态对。Q_{ndiag} 的计算可以通过两个步骤完成。首先, 识别 G_v 中的所有不可控模糊循环。接着, 纳入可以通过仅由不可控事件组成的运行到达这些循环的状态。这个过程的多项式复杂度直接源于 G_v 中变迁的多项式数量[126]。对于每个模糊循环, 可以通过在其各自最近的可诊断状态对中应用控制动作来在 G_v 中移除不可控模糊循环, 或通过对其中任一可控事件应用控制动作来避免可控模糊循环的存在。

定义 7.5. 对于任何可观时间字符串 $w \in P(L(G))$, 所有 w**一致状态** 的集合定义为

$$C(w) = \{q \in Q \mid \exists w' \in L(G), \exists q_0 \in Q_i : (q_0, w', q) \in \Delta^*, P(w') = w\}.$$

通俗地说, 状态 q 与观测 w 一致, 如果存在一个时间字符串 w' 导向状态 q, 并且 w' 的投影与 w 相符。

根据上一节对可诊断性验证的分析, 一个不可诊断的 TIA 可能会进入某些模糊循环并永久停留在其中。为了防止这种情况发生, 应在离线阶段对以下两类模糊循环进行预处理：

1. 对于每一个可控的模糊循环, 选择其中任意一个可控的变迁, 然后限制在G中相应变迁对的权重, 从而防止在G_v中形成模糊循环。

2. 对于每一个不可控的模糊循环, 限制从可诊断状态对到不可诊断对的可控变迁对应的每一对变迁的权重, 确保没有可达的不可诊断状态对。

需要注意的是, 上述限制操作实际上是在在线阶段实施的。特别是, 在某些情况下, 如果限制操作无法提供解决方案, 则会采取禁用操作作为最后手段。

算法 7.2 离线阶段：预计算待控制的变迁

输入: 一个不可诊断的 TIA $G = (Q, \Sigma, Q_i, \Delta, \mu)$。
输出: 一个待控制变迁的集合 T.

1: 调用算法 7.1 构造验证器 $G_v = (Q_v, \Sigma_v, Q_v^i, \Delta_v, \mu_v)$。
2: 计算在 G_v 中不可诊断状态对的集合 Q_{ndiag}。
3: **if** $Q_v^i \cap Q_{ndiag} \neq \emptyset$ **then**
4:　　**输出:** 无可用的监督器。
5: 设 $T = \emptyset$。
6: **for** 每一个 $q_v' \in Q_{ndiag}$ 和 $\sigma \in \Sigma_c$, 满足 $(q_v, \sigma, q_v') \in \Delta_v$ 且 $q_v \in Q_v \setminus Q_{ndiag}$ **do**
7:　　$T = T \cup \{(q_v, \sigma, q_v')\}$。
8: **for** 在 G_v 中的每一个可控模糊循环中 **do**
9:　　选择任意一个可控变迁 (q_v, σ, q_v') (即 $\sigma \in \Sigma_c$) , 令 $T = T \cup \{(q_v, \sigma, q_v')\}$。

现在描述用于主动诊断监督器设计的离线阶段的算法 7.2, 步骤1构造G的验证器G_v, 并在后续阶段使用它来搜索导致系统不可诊断的变迁。步骤2计算不可诊断状态对集合Q_{ndiag}。步骤3–4描述了初步评估, 其中G_v中存在初始不可诊断状态对意味着不可诊断性的发生无法预防。步骤5 初始化要控制的变迁集T, 该集合用于记录那些将被禁用或其权重将被限制的变迁。步骤6–7和8–9分别处理不可控和可控模糊循环, 以便确保在算法 7.3 中所有不可诊断状态对都是不可达的。

例 7.5. 考虑图 7.1 中的TIA G及其验证器G_v (如图 7.4 所示) , 其中可控事件为$\Sigma_c = \{a, c\}$, 不可控事件为$\Sigma_{uc} = \{b, u, e_f\}$。根据关于 (不) 可控模糊循环的定义 7.3, 可知$(5N, 6F) \xrightarrow{b/[2,2]} (5N, 6F)$是一个不可控模糊循环, 而$(3N, 4F) \xrightarrow{c/[2,2]} (1N, 2F) \xrightarrow{b/[1,2]} (3N, 4F)$是一个可控模糊循环。根据算法 7.2 的步骤2, 不可诊断状态对的集合为$Q_{ndiag} = \{5N6F, 5N5N\}$。由于$Q_v^i = \{0N0N\}$且$Q_v^i \cap Q_{ndiag} = \emptyset$, 因此存在一个可用的监督器。根据步骤6–7, 设$q_v' = (5N, 5N)$, $\sigma = a$, 且$q_v = (3N, 3N)$, 则变迁$(3N3N, a, 5N5N)$被加入集合T。步骤8–9选择变迁$(3N4F, c, 1N2F)$并将其加入T中。

接下来描述主动诊断监督器设计的在线阶段。在此阶段, 区分集合T中的哪些

变迁可以限制权重以及哪些变迁只能被禁用至关重要。首要任务是确定哪些变迁值得进行权重限制, 因为限制操作可以尽可能保留原系统的行为。

算法 7.3 在线阶段：主动诊断监督器设计

输入: TIA $G = (Q, \Sigma, Q_i, \Delta, \mu)$;
观测自动机 $G_o = (Q_o, \Sigma_o, Q_o^i, \Delta_o, \mu_o)$;
被控变迁集 T.
输出: 在线的监督器 $S : P(L(G)) \to \Gamma$.

1: 设观测为 $w = \varepsilon$.
2: 计算 w一致状态集 $C(w)$.
3: 令 $S(w) = \{(x, \sigma, y, \mu(x, \sigma, y)) \mid \exists (x, \sigma, y) \in \Delta \cap (C(w) \times \Sigma \times Q)\}$.
4: **for** 每个 $(((p, l_p), (q, l_q)), \sigma, ((p', l'_p), (q', l'_q))) \in T$ **do**
5: **if** $\{p, q\} \subseteq C(w)$ **then**
6: $\Delta_\sigma = \Delta \cap (C(w) \times \sigma \times Q) = \{(x_1, \sigma, y_1), (x_2, \sigma, y_2), \ldots, (x_M, \sigma, y_M)\}$①;
$X_\nu = \{q_1 \in C(w) \mid Q_o \cap (\{q_1\} \times \{N, F\}) \neq \emptyset\}$;
$I_\nu = \bigcup_{(x,\nu,y) \in X_\nu \times \Sigma_{uo}^* \times \{q_1 \in Q \mid (q_1, \sigma, q_1') \in \Delta_\sigma\}} \mu(x, \nu, y)$②;
if $p' = q'$ **then**
$Q_{mid}^\wedge = \{r \in Q \mid \nu_1, \nu_2 \in (\Sigma_{uo})^* \setminus \{\varepsilon\} : (p, \nu_1, r), (q, \nu_2, r), (r, \sigma, p') \in \Delta^*\}$.
else
$Q_{mid}^\wedge = \emptyset$.
7: 计算下述方程的解 $i_\sigma^m (m = 1, \ldots, M)$③:

$$\begin{cases} i_\sigma^m \subseteq \mu(x_m, \sigma, y_m), \\ \bigcap_{m=1}^M (I_\nu + \mu(x_m, \sigma, y_m)) = \emptyset; \end{cases} \quad (x_m, \sigma, y_m) \in \Delta_\sigma \tag{7-1}$$

8: **if** $(Q_{mid}^\wedge \neq \emptyset) \vee (Q_{mid}^\wedge = \emptyset \wedge p = q \wedge p' = q') \vee$ (解 i_σ^m 不存在) **then**
9: 令 $S(w) = S(w) \setminus \{(x, \sigma, y, I) \mid (x, \sigma, y) \in \Delta_\sigma, I \in 2^{\mathbb{R}_{\geq 0}} \setminus \{\emptyset\}\}$.
10: **else**
11: **for** $m = 1, \ldots, M$ **do**
12: **if** $(x_m, \sigma, y_m, I_m) \in S(w)$ 且 $I_m \in 2^{\mathbb{R}_{\geq 0}} \setminus \{\emptyset\}$ **then**
13: $S(w) = S(w) \setminus \{(x_m, \sigma, y_m, I_m)\} \cup \{(x_m, \sigma, y_m, i_\sigma^m)\}$.
14: 等待下一个事件观测 (σ, t), 然后令 $w = w(\sigma, t)$.
15: 回到步骤 2.

算法 7.3 概述了设计主动诊断监督器的在线过程。此算法中基于观测来应用控制操作。更具体地说, 给定观测w后, 首先计算当前状态估计$C(w)$和初始控制操作$S(w)$。从步骤4开始, 区分T中可以限制权重的变迁和只能被禁用的变迁, 然后应用相应的控制操作。步骤6计算在G中标记为σ的所有从$C(w)$中的状态出发的变迁集合, 并将其记为Δ_σ。该集合包含所有需要被禁用 (在步骤9中) 或被限制权重

① 定义Δ_σ的元素从1到M按顺序索引。
② 特别地, 如果$X_\nu \times \Sigma_{uo}^* \times \{q \in Q \mid (q, \sigma, q') \in \Delta_\sigma\} = \emptyset$, 则定义$I_\nu = [0, 0]$; 如果$|X_\nu \times \Sigma_{uo}^* \times \{q \in Q \mid (q, \sigma, q') \in \Delta_\sigma\}| = 1$, 则$I_\nu = \mu(x, \nu, y)$。
③ 这实际上是一个约束满足问题 (Constraint Satisfaction Problem) , 可以使用现有算法进行求解[143]。

(在步骤13中) 的变迁。当前状态估计$C(w)$中具有可观输入变迁 (或以σ标记的可观输出变迁) 的状态集合记为X_ν (或$\{q_1 \in Q \mid (q_1, \sigma, q_1') \in \Delta_\sigma\}$) 。符号$I_\nu$表示标记为$\nu \in \Sigma_{uo}^*$的变迁的权重区间, 这些变迁从$X_\nu$中的状态到$\{q_1 \in Q \mid (q_1, \sigma, q_1') \in \Delta_\sigma\}$中的状态。当$T$中的变迁 (参见步骤4) 满足$p' = q'$时, 计算可以通过不可观路径从$p$和$q$**均**到达的中间状态集合$Q_{mid}^{\wedge}$; 否则, 它是一个空集。这个计算是为了区分需要限制或禁用的变迁。如果$Q_{mid}^{\wedge} \neq \emptyset$或 ($Q_{mid}^{\wedge} = \emptyset$, $p = q$且$p' = q'$) , 那么必须应用禁用操作。步骤7将新的区间权重分配表述为一组不等式方程进行求解。其目的是确保每个变迁的新分配权重不会相互重叠。特别地, 考虑到物理可实现性, 要求任何新分配的权重应为其原始权重的子区间, 而不是引入不属于原系统的新权重。当方程组无解时, 也必须应用禁用操作。禁用过程在步骤8–9中执行。如果变迁$(x, \sigma, y, \mu(x, \sigma, y))$不在$S(w)$中, 则意味着此变迁在之前的步骤9中已被禁用。步骤12中的检验避免了错误启用被禁用的变迁。

备注 7.1. 在此讨论步骤8中析取范式的含义, 参考图 7.5。当$Q_{mid}^{\wedge} \neq \emptyset$ (左) 时, 若变迁$(((p, l_p), (q, l_q)), \sigma, ((p', l_p'), (q', l_q')))$的形成是由于$\mu(p, \nu_1\sigma, p') \cap \mu(q, \nu_2\sigma, q') \neq \emptyset$, 那么无论如何限制权重, 都无法防止这种变迁在验证器中形成。类似的分析适用于$Q_{mid}^{\wedge} = \emptyset \wedge p = q \wedge p' = q'$的情况 (右) 。

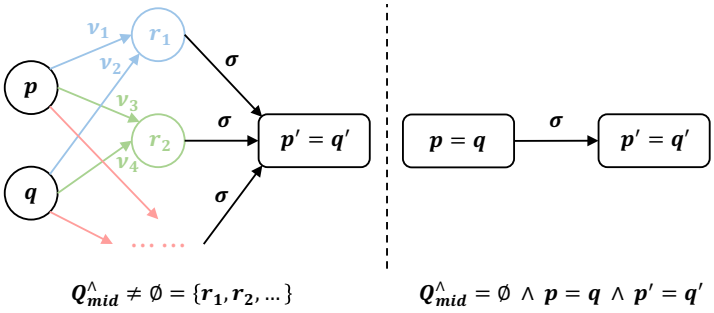

图 **7.5** 析取范式的解释草图

定理 7.2. 算法 7.3 设计的监督器S可以使不可诊断的 G 的闭环系统S/G可诊断。

证明. 根据定理 7.1, 如果G不可诊断, 则在G_v中存在至少一个模糊循环

$$v_1 \xrightarrow{\sigma_1/I_1} v_2 \cdots \xrightarrow{\sigma_{k-1}/I_{k-1}} v_k \xrightarrow{\sigma_k/I_k} v_1,$$

其中$v_i = (p_iN, q_iF)$, $i = 1, 2, \ldots, k$。

若循环可控, 根据定义 7.3, 可以找到可控变迁 $(p_jN, q_jF) \xrightarrow{\sigma_j/I_j} (p_{j+1}N, q_{j+1}F)$。根据算法 7.2 的步骤8–9, 有 $\big((p_jN, q_jF), \sigma_j, (p_{j+1}N, q_{j+1}F)\big) \in T$。对于变迁 $\big((p_jN, q_jF),$

$\sigma_j, (p_{j+1}N, q_{j+1}F)\big) \in \Delta_v$, 在$G$中可能存在以下变迁：

$$p_j \xrightarrow{\sigma_j/I'_j} p_{j+1} \text{ 和 } q_j \xrightarrow{\sigma_j/I''_j} q_{j+1} \tag{7-2}$$

其中 $I'_j \cap I''_j = I_j$, 和 (或)

$$p_j \xrightarrow{\nu'/I'_\nu} r' \xrightarrow{\sigma_j/I'_{\sigma_j}} p_{j+1} \text{ 和 } q_j \xrightarrow{\nu''/I''_\nu} r'' \xrightarrow{\sigma_j/I''_{\sigma_j}} q_{j+1} \tag{7-3}$$

其中 $(I'_\nu + I'_{\sigma_j}) \cap (I''_\nu + I''_{\sigma_j}) = I_j$ 且 $\nu', \nu'' \in \Sigma^*_{uo} \backslash \{\varepsilon\}$。当同时满足情况 (7-2) 和 (7-3) 时, 根据算法 7.3 的步骤6, 有 (p_j, σ_j, p_{j+1}), (q_j, σ_j, q_{j+1}), (r', σ_j, p_{j+1}), $(r'', \sigma_j, q_{j+1}) \in \Delta_\sigma$ 并且 $I'_\nu \cup I''_\nu \subseteq I_\nu$[①]。如果 $p_{j+1} \neq q_{j+1}$, 意味着 $Q^{\wedge}_{mid} = \emptyset$, 那么必然存在不同的变迁对 (r', σ_j, p_{j+1}) 和 (r'', σ_j, q_{j+1}), 以及 (或) (p_j, σ_j, p_{j+1}) 和 (q_j, σ_j, q_{j+1})。因此, 可以限制它们的权重以防止不可诊断性。通过步骤7和11–13, 为每个当前可能的变迁 $(x^m, \sigma, y^m) \in \Delta_\sigma$ 分配一个新的权重 i^m_σ。根据公式 (7-1), 每个新分配的权重 i^m_σ 应满足方程 $(I_\nu + i^m_\sigma) \cap \cdots \cap (I_\nu + i^M_\sigma) = \emptyset$, 以确保在$G_v$中移除变迁 (v_j, σ_j, v_{j+1})。另一方面, 不会生成其他新的变迁, 因为任何新分配的权重都是其原始权重的子区间。因此, 这个可控循环在闭环系统 $(S/G)_v$ 的验证器中无法形成。由定理 7.1 可知, S/G 是可诊断的。如果 $p_{j+1} = q_{j+1}$ 且 $Q^{\wedge}_{mid} \neq \emptyset$, 则通过步骤8–9禁用所有从当前状态估计开始并标记为 σ_j 的可能变迁。因此, 这个可控循环也无法形成。如果 $p_{j+1} = q_{j+1}$ 但 $Q^{\wedge}_{mid} = \emptyset$, 则意味着不存在变迁 (p_j, ν', r') 和 (q_j, ν'', r'') 满足 $r' = r''$。因此, 变迁对 (r', σ_j, p_{j+1}) 和 (r'', σ_j, p_{j+1}) 仍然存在且 $r' \neq r''$。然而, 对于 (p_j, σ_j, p_{j+1}) 和 (q_j, σ_j, p_{j+1}), 如果 $p_j = q_j$, 这两条变迁本质上是同一条, 因此必须应用禁用操作。否则, 如果 $p_j \neq q_j$, 则 (r', σ_j, p_{j+1}) 和 (r'', σ_j, p_{j+1}) 以及 (p_j, σ_j, p_{j+1}) 和 (q_j, σ_j, p_{j+1}) 都是不同的变迁对, 限制操作可以生效。当仅满足情况 (7-2) 或 (7-3) 时, 证明类似于上述情况, 故省略。

如果循环不可控, 则在进入循环前应用控制操作。根据算法 7.2 的步骤2, 有 $v_1, v_2, \ldots, v_k \in Q_{ndiag}$。如果存在变迁序列 $q_c \xrightarrow{\sigma_c/I_c} q_{uc} \xrightarrow{w_{uc}/I_{uc}} v_j$, 其中 $\sigma_c \in \Sigma_c$ 且 $w_{uc} \in \Sigma^*_{uc}$, 则将所有变迁 $(q_{uc}, w_{uc}, v_j) \in \Delta^*$ 经过的状态纳入 Q_{ndiag}。之后根据上述的类似情况应用禁用或限制操作。其余证明与前述类似, 故省略。 □

当任何限制操作被替换为禁用操作时, 监督器仍然可以使闭环系统可诊断, 但往往会导致过于严格的行为。这一现象突显了所提出方法的灵活性。

① 可能存在不止两条这样的变迁, 即 $I_\nu = I'_\nu \cup I''_\nu \cup \cdots$。简单起见, 不失一般性地, 这里仅考虑两条变迁。

例 7.6. 考虑算法 7.3 中的在线阶段。图 7.6 描绘了闭环系统 $(S/G)_v$ 的验证器。初始时, 计算 $C(\varepsilon)=\{0\}$。对于观测 $(a,1.5)$, 得到 $(a,1.5)$一致状态集为 $C((a,1.5))=\{1\}$。在观测到 $w=(a,1.5)(b,3.5)$ 后, 一致状态集变为 $C((a,1.5)(b,3.5))=\{3,4\}$。在步骤3中, 初始控制操作 $S(w)$ 如下：

$$S(w)=\{(3,a,5,[0,1]),(3,c,1,[2,4]),(4,a,4,[2,3]),(4,c,2,[1,2])\}.$$

对于变迁 $(3N3N,a,5N5N)\in T$, 根据步骤6, 有 $\Delta_\sigma=\{(3,a,5),(4,a,4)\}$ 和 $Q^{\wedge}_{mid}=\emptyset\wedge p=q=3\wedge p'=q'=5$。因此, 通过步骤8–9 禁用事件 a, 得到：

$$S(w)=\{(3,c,1,[2,4]),(4,c,2,[1,2])\}.$$

对于变迁 $(3N4F,c,1N2F)\in T$, 有 $\Delta_\sigma=\{(3,c,1),(4,c,2)\}$, $X_\nu=C(w)=\{3,4\}$, $I_\nu=[0,0]$, 且 $Q^{\wedge}_{mid}=\emptyset$。因此, 根据步骤7 和 11–13 重新分配 $(3,c,1)$ 和 $(4,c,2)$ 的权重, 其中选择了两个满足方程的解 $[2,4]$ 和 $[1,1.5]$。

$$[2,4]\subseteq\mu(3,c,1)=[2,4],\ [1,1.5]\subseteq\mu(4,c,2)=[1,2],$$
$$([0,0]+[2,4])\cap([0,0]+[1,1.5])=\emptyset$$

因此有 $S(w)=\{(3,c,1,[2,4]),(4,c,2,[1,1.5])\}$, 这导致在 $(S/G)_v$ 中缺少变迁 $(3N4F,c,1N2F)$, 参见图 7.6。因此根据定理 7.1, S/G 是可诊断的。

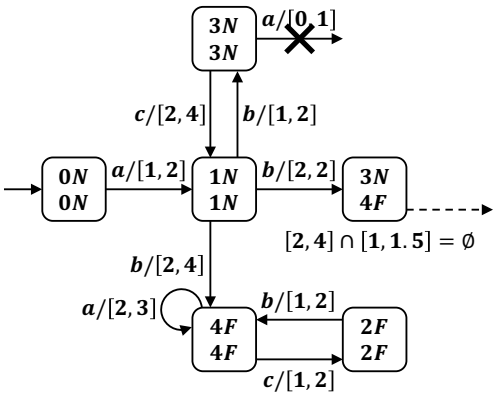

图 **7.6** 闭环系统 $(S/G)_v$ 的验证器**,** 叉号表示变迁被禁用**,** 虚线表示因权重修改被移除

7.3 案例研究：智能家居系统

本节通过一个智能家居系统的案例分析展示所提方法的适用性。如图 7.7 所示, 系统包含一个带充电点的扫地机器人、一个远程智能操作终端 (智能手机) 以及两个带有 Wi-Fi 的卧室。其TIA 模型 G' 见图 7.8。

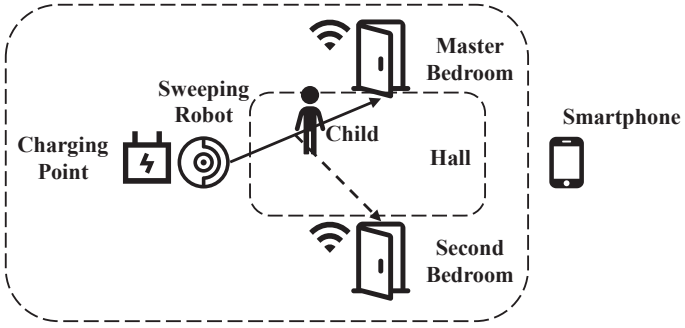

图 **7.7** 智能家居系统

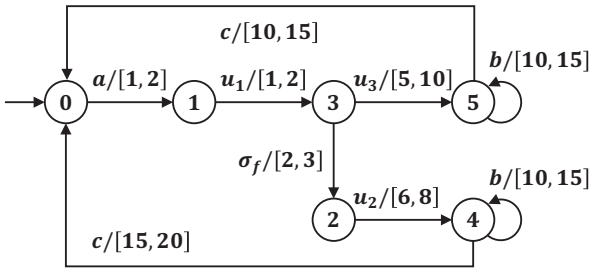

图 **7.8** 图 **7.7** 中智能家居系统的 **TIA** 模型 G'

假设当前任务是让机器人清扫主卧 (若有需要, 可对次卧进行类似建模) , 系统的运行过程如下：最初, 机器人停留在充电点, 等待任务分配 (状态 0) 。随后, 智能手机给机器人分配清扫主卧的任务 (事件 a) 。尽管仍在充电点, 机器人已接收到清扫任务 (状态 1) 。接着, 机器人从充电点出发 (事件 u_1) 。状态 3 为中间状态, 表示机器人当前在走廊中。状态 3 后有两个可能的事件：事件 u_3 表示机器人持续移动, 顺利到达主卧 (状态 5) ； 事件 e_f 表示机器人意外被小孩踢到, 导致延误及偏航。被踢后, 机器人继续向前移动 (事件 u_2) , 但因方向变化误入次卧 (状态 4) 。接下来, 机器人开始清扫并定期向智能手机发送清扫通知 (事件 b) 。最后, 在完成清扫工作后, 机器人返回充电点并通知智能手机 (事件 c) 。

在该系统中, 机器人的故障和移动被视为不可观的, 因为用户无需在家, 即 $\Sigma'_{uo} = \{u_1, u_2, u_3, e_f\}$ 且 $\Sigma'_f = \{e_f\}$。唯一可控事件是 b, 因为通知频率依赖于卧室中的 Wi-Fi 信号强度, 这可以通过智能手机进行调整, 即 $\Sigma'_c = \{b\}$。

现在根据算法 7.1 为 TIA 模型 G' 构造验证器 G'_v, 并应用定理 7.1 检验其可诊断性。如果 G'_v 不可诊断, 则应用算法 7.2 和 7.3 设计相应的控制策略以强制其可诊断性。观测自动机 G'_o 和验证器 G'_v 如图 7.9 和 7.10 所示。存在两个可控的模糊循环 $(5N,4F) \xrightarrow{b/[10,15]} (5N,4F)$ 和 $(5N,4F) \xrightarrow{c/[15,15]} (0N,0F) \xrightarrow{a/[1,2]} (1N,1F) \xrightarrow{b/[19,27]} (5N,4F)$, 且仅有可控事件 b, 这表明 TIA G' 是不可诊断的。因此需要应用算法 7.2 和 7.3。根据算法 7.2 的步骤 5–9, 得到

$$T = \{((5N,4F),b,(5N,4F)),((1N,1F),b,(5N,4F))\}.$$

在线阶段中, 对于观测 $w=(a,2)(b,29)$, 得到 $C(w)=X_\nu=\{4,5\}$ 及

$$S(w)=\{(5,c,0,[10,15]),(5,b,5,[10,15]),(4,c,0,[15,20]),(4,b,4,[10,15])\}.$$

对于变迁 $((5N,4F),b,(5N,4F)) \in T$, 有 $\Delta_\sigma=\{(4,b,4),(5,b,5)\}$, $I_\nu=[0,0]$, 及 $Q^{\wedge}_{mid}=\emptyset$。通过智能手机增强主卧的 Wi-Fi 信号并减弱次卧的信号, 从而分别增加和减少机器人发送消息的频率。具体地, 对 $S(w)$ 进行如下限制：

$$\begin{aligned} S(w) = S(w) \setminus \{&(5,b,5,[10,15]),(4,b,4,[10,15])\} \\ &\cup \{(5,b,5,[10,12]),(4,b,4,[13,15])\}, \end{aligned}$$

满足 $[10,12]\subseteq[10,15],\ [13,15]\subseteq[10,15], ([0,0]+[10,12])\cap([0,0]+[13,15])=\emptyset$. 因此, 如果手机每隔 10 到 12 秒收到一次来自机器人的消息, 则可以推断机器人已成功到达主卧并正在进行打扫。反之, 如果消息每隔 13 到 15 秒接收一次, 则表明机器人发生了故障, 导致其错误地进入次卧进行打扫。

对于观测 $w=(a,2)(b,29)(c,44)(a,46)$, 有 $C(w)=\{1,2,3,4,5\}$, $X_\nu=\{1,4,5\}$ 及 $S(w)=\{(5,c,0,[10,15]),(5,b,5,[10,15]),(4,c,0,[15,20]),(4,b,4,[10,15])\}$. 对于变迁 $((5N,4F),b,(5N,4F))\in T$, 有 $\Delta_\sigma=\{(4,b,4),(5,b,5)\}$,

$$I_\nu=([1,2]+[5,10])\cup([1,2]+[2,3]+[6,8])=[6,13],$$

及 $Q^{\wedge}_{mid}=\emptyset$。然而, 对于这样的 Δ_σ 和 I_ν, 无法找到方程 (7-1) 的解。因此, 必须对事件 b 应用禁用操作, 从而得到

$$\begin{aligned} S(w) &= S(w) \setminus \{(5,b,5,[10,15]),(4,b,4,[10,15])\} \\ &= \{(5,c,0,[10,15]),(4,c,0,[15,20])\}. \end{aligned}$$

对于变迁 $((1N,1F),b,(5N,4F))\in T$, 有与变迁 $((5N,4F),b,(5N,4F))$ 相同的 Δ_σ、I_ν 和 $Q^{\wedge}_{mid}$。因此, 保持控制操作 $S(w)$ 不变。

以观测 $w=(a,2)(b,29)$ 为例。在这种情况下, 虽然无法知道系统当前状态是 4 还是 5, 对于相同的观测 $w=(a,\tau_1)(b,\tau_2)$, 其中 $\tau_1\in[1,2]$ 和 $\tau_2\in[20,29]$, 因此总是采用一致的控制策略：

$$S(w)=\{(5,c,0,[10,15]),(5,b,5,[10,12]),(4,c,0,[15,20]),(4,b,4,[13,15])\},$$

利用系统的时间信息来实现可诊断性的使能。

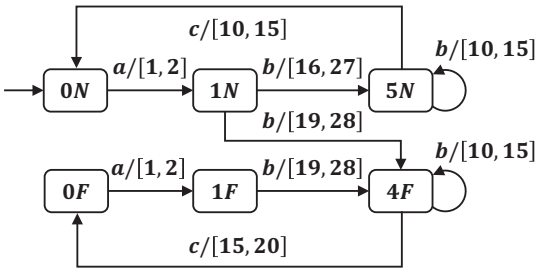

图 **7.9** 图 **7.8** 中 G' 的观测自动机 G'_o

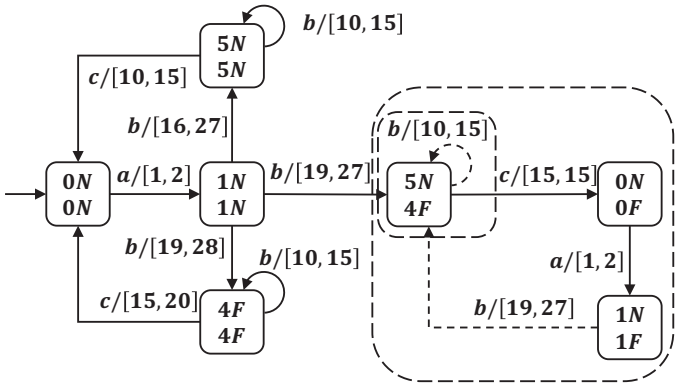

图 **7.10** 验证器 G'_v 的一部分**,** 其中状态 $(4F,5N)$ 及其连接的变迁被省略**,** 模糊循环由虚线框出

7.4 本章小结

本章首先构建了用于 TIA 可诊断性验证的验证器, 进而提出了一种结合权重限制和事件禁用的新型混合控制策略, 用于 TIA 的主动诊断。该策略优先通过权重限制保持系统原始行为, 仅在必要时采用事件禁用。最后, 通过智能家居系统案例验证了所提方法的有效性。

第8章　总结与展望

本章对全文研究内容做总结, 并对未来工作进行展望。

8.1　总结

本文从多维度对离散事件系统 (DES) 的安全性进行了系统性研究, 包括不透明性、临界可观测性、故障诊断/预测等关键问题, 并在模块化与定时框架下提出了一系列创新的性质验证与控制器综合方法。

在增强型IFO研究方面, 提出了 PSIFO 和 SIFO 两种新型安全性质及其验证方法。这些性质通过并发组合方法进行验证, 虽然面临较高的计算复杂度, 但通过引入有效的简化条件显著提升了验证效率。与现有文献中的标准 IFO 相比, PSIFO 不仅提供了更强的保密性保证, 同时具有更低的验证复杂度, 展现出重要的理论价值和实际应用潜力。

临界可观测性 (CO) 研究取得了重要突破。通过建立基于语言的CO与基于状态的CO在DFA框架下的等价关系, 以及MOC条件, 首次实现了模块化系统中针对局部和全局规范的非阻塞、最大许可、可控、常态且临界可观的监督器综合, 为解决大规模系统的安全性保障问题提供了有效方案。

在故障诊断方面, 对单体NFA、模块化NFA和无界LPN系统的弱可诊断性与弱可预测性进行了系统的复杂度分析。研究结果表明: 在单体NFA中这些问题具有 PSPACE-completeness; 在模块化NFA中达到 EXPSPACE-completeness; 而在无界 LPN 中, 弱可预测性被证明是不可判定的, 弱可诊断性则属于Ackermann-hard问题且可判定性尚未解决。这些发现填补了故障诊断领域复杂度分析的空白, 为实际系统设计提供了重要的理论依据。

针对定时系统, 基于CTA模型提出了创新的时间并行组合方法, 并开发了验证CTA协同可诊断性的有效算法。通过引入约简条件, 成功降低了CTA可诊断性的验证复杂度。特别地, 研究揭示了CTA框架下协同可诊断性与集中可诊断性的等价关系, 这一发现与FSA框架下的结论形成鲜明对比。

进一步在TIA框架下, 构建了可诊断性验证器并提出了结合权重限制和事件禁用的主动诊断控制策略。通过智能家居系统的案例研究, 验证了该方法在实际应用中的有效性。

8.2 展望

本研究体系具有广阔的拓展空间和深入研究的价值, 未来工作将重点关注以下方向:

首先, 针对SIFO验证的效率瓶颈问题, 将开发更高效的验证算法并探索其复杂度边界[144-145]。同时, 研究增强IFO的控制问题具有重要的理论意义[146]。

临界可观测性研究将向多智能体DES[147]、网络化DES[148-149]以及面临网络攻击的DES[150-151]等更复杂系统拓展[152]。探索将不透明性[19]和临界可观测性研究扩展到定时系统框架也值得关注。

针对无界LPN的弱可预测性不可判定性, 将致力于寻找最大可判定子类[57-58]并研究其验证方法[62]。同时, 将继续探究弱可诊断性的可判定性问题。研究 NFA 模型中弱可诊断性与弱可预测性之间的归约关系也颇具理论价值。

定时框架下的研究将扩展到部分观测条件下的集中式和分散式监督控制理论。重点探索加权自动机框架中可观测性与可诊断性之间的深层关联[153]。

此外, 将深入研究加权自动机的协同或模块化可诊断性的强制实施方法[43], 优化在线算法中的限制操作和事件控制策略。针对更复杂的系统行为, 研究可控事件的最优选择策略[154-155], 以实现可诊断TIA的最大许可行为。

最后, 将突破传统故障事件分析方法的局限, 在定时框架下探索基于故障模式[156-157]的故障诊断/预测新方法。

References

[1] Baheti R, Gill H. Cyber-physical systems[J]. The Impact of Control Technology, 2011, 12(1): 161-166.

[2] Cassandras C G, Lafortune S. Introduction to discrete event systems[M]. Springer, 2021.

[3] Franklin G F, Powell J D, Emami-Naeini A, et al. Feedback control of dynamic systems[M]. Prentice Hall, 2002.

[4] Brogan W L. Modern control theory[M]. Pearson Education, 1985.

[5] Yin X. Estimation and verification of partially-observed discrete-event systems[J]. Wiley Encyclopedia of Electrical and Electronics Engineering, 2019.

[6] Wonham W M, Cai K. Supervisory control of discrete-event systems[M]. Springer, 2022.

[7] Jacob R, Lesage J J, Faure J M. Overview of discrete event systems opacity: Models, validation, and quantification[J]. Annual Reviews in Control, 2016, 41: 135-146.

[8] Lafortune S, Lin F, Hadjicostis C N. On the history of diagnosability and opacity in discrete event systems[J]. Annual Reviews in Control, 2018, 45: 257-266.

[9] Basilio J C, Hadjicostis C N, Su R. Analysis and control for resilience of discrete event systems: Fault diagnosis, opacity and cyber security[J]. Foundations and Trends® in Systems and Control, 2021, 8(4): 285-443.

[10] Lin F. Opacity of discrete event systems and its applications[J]. Automatica, 2011, 47(3): 496-503.

[11] Luo Y, Miao S, Lan W, et al. Supervisory control for current-state e-opacity enforcement under encrypted observations[C]//2025 37th Chinese Control and Decision Conference (CCDC). IEEE, 2025.

[12] Miao S, Cui B, Ji Y, et al. Protect your knowledge: Epistemic property enforcement of discrete event systems with asymmetric information[J]. IEEE Control Systems Letters, 2025, under review.

[13] Mazaré L. Using unification for opacity properties[J]. Proceedings of the 4th IFIP WG1, 2004, 7: 165-176.

[14] Saboori A, Hadjicostis C N. Notions of security and opacity in discrete event systems[C]//2007 46th IEEE Conference on Decision and Control (CDC). IEEE, 2007: 5056-5061.

[15] Saboori A, Hadjicostis C N. Verification of initial-state opacity in security applications of discrete event systems[J]. Information Sciences, 2013, 246: 115-132.

[16] Saboori A, Hadjicostis C N. Verification of k-step opacity and analysis of its complexity[J]. IEEE Transactions on Automation Science and Engineering, 2011, 8(3): 549-559.

[17] Saboori A, Hadjicostis C N. Verification of infinite-step opacity and complexity considerations [J]. IEEE Transactions on Automatic Control, 2011, 57(5): 1265-1269.

[18] Wu Y C, Lafortune S. Comparative analysis of related notions of opacity in centralized and

coordinated architectures[J]. Discrete Event Dynamic Systems, 2013, 23(3): 307-339.

[19] Shen L, Miao S, Lai A, et al. Verification of initial-and-final-state opacity for unambiguous weighted automata[J]. ISA transactions, 2024, 148: 237-246.

[20] Dulce-Galindo J A, Alves L V R, Raffo G V, et al. Enforcing state-based opacity using synchronizing automata[C]//2021 60th IEEE Conference on Decision and Control (CDC). IEEE, 2021: 7009-7014.

[21] Balun J, Masopust T. Comparing the notions of opacity for discrete-event systems[J]. Discrete Event Dynamic Systems, 2021, 31(4): 553-582.

[22] Yin X, Lafortune S. A new approach for the verification of infinite-step and k-step opacity using two-way observers[J]. Automatica, 2017, 80: 162-171.

[23] Lan H, Tong Y, Guo J, et al. Comments on "A new approach for the verification of infinite-step and k-step opacity using two-way observers"[automatica 80 (2017) 162–171][J]. Automatica, 2020, 122: 109290.

[24] Wintenberg A, Blischke M, Lafortune S, et al. A general language-based framework for specifying and verifying notions of opacity[J]. Discrete Event Dynamic Systems, 2022, 32(2): 253-289.

[25] Hadjicostis C N. Estimation and inference in discrete event systems: A model-based approach with finite automata[M]. Springer, 2020.

[26] Miao S, Lai A, Yu X, et al. Verification of detectability for unambiguous weighted automata using self-composition[C]//2023 9th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, 2023: 251-256.

[27] Xu G, Miao S, Lai A, et al. Verification of delayed detectability for unambiguous weighted automata[C]//2023 6th International Conference on Mechatronics, Robotics and Automation (ICMRA). IEEE, 2023: 1-5.

[28] Shen L, Miao S, Lai A, et al. Initial-and-final-state detectability of nondeterministic finite-state automata[C]//2024 14th Asian Control Conference (ASCC). IEEE, 2024: 509-514.

[29] Miao S, Lai A, Komenda J, et al. Timed initial-state detectability of discrete-event systems by algebraic method[J]. Nonlinear Analysis: Hybrid Systems, 2025, under review.

[30] Falcone Y, Marchand H. Enforcement and validation (at runtime) of various notions of opacity [J]. Discrete Event Dynamic Systems, 2015, 25: 531-570.

[31] Ma Z, Yin X, Li Z. Verification and enforcement of strong infinite-and k-step opacity using state recognizers[J]. Automatica, 2021, 133: 109838.

[32] Han X, Zhang K, Zhang J, et al. Strong current-state and initial-state opacity of discrete-event systems[J]. Automatica, 2023, 148: 110756.

[33] Balun J, Masopust T. Verifying weak and strong k-step opacity in discrete-event systems[J]. Automatica, 2023, 155: 111153.

[34] Han X, Zhang K, Li Z. Verification of strong k-step opacity for discrete-event systems[C]// 2022 61st IEEE Conference on Decision and Control (CDC). IEEE, 2022: 4250-4255.

[35] Chu Q, Wei J, Han X, et al. Transformations between opacity for discrete-event systems[C]//

2022 41st Chinese Control Conference (CCC). IEEE, 2022: 1611-1616.

[36] Wonham W M, Ramadge P J. Modular supervisory control of discrete-event systems[J]. Mathematics of Control, Signals and Systems, 1988, 1(1): 13-30.

[37] De Queiroz M H, Cury J E R. Modular supervisory control of large scale discrete event systems [M]//Discrete Event Systems: Analysis and Control. Springer, 2000: 103-110.

[38] Lin F, Wonham W M. On observability of discrete-event systems[J]. Information Sciences, 1988, 44(3): 173-198.

[39] Cho H, Marcus S I. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation[J]. Mathematics of Control, Signals and Systems, 1989, 2(1): 47-69.

[40] Brandt R D, Garg V, Kumar R, et al. Formulas for calculating supremal controllable and normal sublanguages[J]. Systems & Control Letters, 1990, 15(2): 111-117.

[41] Takai S, Ushio T. Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control[J]. Systems & Control Letters, 2003, 49(3): 191-200.

[42] Komenda J, Masopust T. Hierarchical supervisory control under partial observation: Normality [J]. IEEE Transactions on Automatic Control, 2023, 68(12): 7286-7298.

[43] Komenda J, Masopust T. Supervisory control of modular discrete-event systems under partial observation: Normality[J]. IEEE Transactions on Automatic Control, 2024, 69(6): 3796-3807.

[44] Pola G, De Santis E, Di Benedetto M D, et al. Design of decentralized critical observers for networks of finite state machines: A formal method approach[J]. Automatica, 2017, 86: 174-182.

[45] Masopust T. Critical observability for automata and Petri nets[J]. IEEE Transactions on Automatic Control, 2020, 65(1): 341-346.

[46] Cong X, Fanti M P, Mangini A M, et al. Critical observability verification and enforcement of labeled Petri nets by using basis markings[J]. IEEE Transactions on Automatic Control, 2023, 68(12): 8158-8164.

[47] Lin F. Diagnosability of discrete event systems and its applications[J]. Discrete Event Dynamic Systems, 1994, 4: 197-212.

[48] Sampath M, Sengupta R, Lafortune S, et al. Diagnosability of discrete-event systems[J]. IEEE Transactions on Automatic Control, 1995, 40(9): 1555-1575.

[49] Jiang S, Huang Z, Chandra V, et al. A polynomial algorithm for testing diagnosability of discrete-event systems[J]. IEEE Transactions on Automatic Control, 2001, 46(8): 1318-1321.

[50] Yoo T S, Lafortune S. Polynomial-time verification of diagnosability of partially observed discrete-event systems[J]. IEEE Transactions on Automatic Control, 2002, 47(9): 1491-1495.

[51] Moreira M V, Jesus T C, Basilio J C. Polynomial time verification of decentralized diagnosability of discrete event systems[J]. IEEE Transactions on Automatic Control, 2011, 56(7): 1679-1684.

[52] Genc S, Lafortune S. Predictability of event occurrences in partially-observed discrete-event

systems[J]. Automatica, 2009, 45(2): 301-311.

[53] Yin X, Lafortune S. Verification complexity of a class of observational properties for modular discrete events systems[J]. Automatica, 2017, 83: 199-205.

[54] Masopust T, Yin X. Complexity of detectability, opacity and A-diagnosability for modular discrete event systems[J]. Automatica, 2019, 101: 290-295.

[55] Cabasino M P, Giua A, Lafortune S, et al. A new approach for diagnosability analysis of Petri nets using verifier nets[J]. IEEE Transactions on Automatic Control, 2012, 57(12): 3104-3117.

[56] Ran N, Su H, Wang S. An improved approach to test diagnosability of bounded Petri nets[J]. IEEE/CAA Journal of Automatica Sinica, 2017, 4(2): 297-303.

[57] Yin X, Lafortune S. On the decidability and complexity of diagnosability for labeled Petri nets [J]. IEEE Transactions on Automatic Control, 2017, 62(11): 5931-5938.

[58] Bérard B, Haar S, Schmitz S, et al. The complexity of diagnosability and opacity verification for Petri nets[J]. Fundamenta Informaticae, 2018, 161(4): 317-349.

[59] Lefebvre D. Fault diagnosis and prognosis with partially observed Petri nets[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2014, 44(10): 1413-1424.

[60] Yin X. Verification of prognosability for labeled Petri nets[J]. IEEE Transactions on Automatic Control, 2017, 63(6): 1828-1834.

[61] You D, Wang S, Seatzu C. Verification of fault-predictability in labeled Petri nets using predictor graphs[J]. IEEE Transactions on Automatic Control, 2019, 64(10): 4353-4360.

[62] Ran N, Hao J, Seatzu C. Prognosability analysis and enforcement of bounded labeled Petri nets[J]. IEEE Transactions on Automatic Control, 2022, 67(10): 5541-5547.

[63] Cao L, Shu S, Lin F, et al. Weak diagnosability of discrete-event systems[J]. IEEE Transactions on Control of Network Systems, 2021, 9(1): 184-196.

[64] Qiu W, Kumar R. Decentralized failure diagnosis of discrete event systems[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2006, 36(2): 384-395.

[65] Wang Y, Yoo T S, Lafortune S. Diagnosis of discrete event systems using decentralized architectures[J]. Discrete Event Dynamic Systems, 2007, 17(2): 233-263.

[66] Tripakis S. Fault diagnosis for timed automata[C]//2022 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT). Springer, 2002: 205-221.

[67] Alur R, Dill D L. A theory of timed automata[J]. Theoretical Computer Science, 1994, 126 (2): 183-235.

[68] Cassez F. The complexity of codiagnosability for discrete event and timed systems[J]. IEEE Transactions on Automatic Control, 2012, 57(7): 1752-1764.

[69] Lai A, Lahaye S, Giua A. State estimation of max-plus automata with unobservable events[J]. Automatica, 2019, 105: 36-42.

[70] Lai A, Lahaye S, Giua A. A two-step approach for fault diagnosis of max-plus automata [C]//2019 6th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, 2019: 1061-1066.

[71] Pin J E. Tropical semirings[M]. Cambridge University Press, 1998.

[72] Komenda J, Lahaye S, Boimond J L, et al. Max-plus algebra in the history of discrete event systems[J]. Annual Reviews in Control, 2018, 45: 240-249.

[73] Lai A, Komenda J, Lahaye S. Diagnosability of unambiguous max-plus automata[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(11): 7302-7311.

[74] Lai A, Lahaye S, Komenda J. Observer construction for polynomially ambiguous max-plus automata[J]. IEEE Transactions on Automatic Control, 2021, 67(3): 1582-1588.

[75] Li J, Lefebvre D, Hadjicostis C N, et al. Observers for a class of timed automata based on elapsed time graphs[J]. IEEE Transactions on Automatic Control, 2021, 67(2): 767-779.

[76] Zhang K, Raisch J. Diagnosability of labeled weighted automata over the monoid ($\mathbb{Q}_{\geq 0}$, +, 0) [C]//2021 60th IEEE Conference on Decision and Control (CDC). IEEE, 2021: 6867-6872.

[77] Rezende C H, Viana G S, Basilio J C. Diagnosability of discrete event systems modeled by time-interval automata[J]. IFAC-PapersOnLine, 2023, 56(2): 8660-8665.

[78] Viana G S, Moreira M V, Basilio J C. Codiagnosability analysis of discrete-event systems modeled by weighted automata[J]. IEEE Transactions on Automatic Control, 2019, 64(10): 4361-4368.

[79] Droste M, Kuich W, Vogler H. Handbook of weighted automata[M]. Springer Science & Business Media, 2009.

[80] Sampath M, Lafortune S, Teneketzis D. Active diagnosis of discrete-event systems[J]. IEEE transactions on Automatic Control, 1998, 43(7): 908-929.

[81] Bertrand N, Fabre E, Haar S, et al. Active diagnosis for probabilistic systems[C]//2014 17th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS). Springer, 2014: 29-42.

[82] Chen K, Miao S, Lai A, et al. Decidability of probabilistic current-state opacity for probabilistic finite automata[C]//2024 43rd Chinese Control Conference (CCC). IEEE, 2024: 1556-1561.

[83] Böhm S, Haar S, Haddad S, et al. Active diagnosis with observable quiescence[C]//2015 54th IEEE Conference on Decision and Control (CDC). IEEE, 2015: 1663-1668.

[84] Hu Y, Ma Z, Li Z. Design of supervisors for partially observed discrete event systems using quiescent information[J]. IEEE Transactions on Automation Science and Engineering, 2024, 21(3): 4778-4789.

[85] Haar S, Haddad S, Melliti T, et al. Optimal constructions for active diagnosis[J]. Journal of Computer and System Sciences, 2017, 83(1): 101-120.

[86] Yin X, Lafortune S. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems[J]. IEEE Transactions on Automatic Control, 2016, 61(8): 2140-2154.

[87] Hu Y, Ma Z, Li Z. Design of supervisors for active diagnosis in discrete event systems[J]. IEEE Transactions on Automatic Control, 2020, 65(12): 5159-5172.

[88] Cao L, Shu S, Lin F. Active fault isolation for discrete event systems[J]. IEEE Transactions on Automatic Control, 2024, 69(8): 4988-5003.

[89] Chen Z, Lin F, Wang C, et al. Active diagnosability of discrete event systems and its application to battery fault diagnosis[J]. IEEE Transactions on Control Systems Technology, 2014, 22(5): 1892-1898.

[90] Lin F, Wang L Y, Chen W, et al. n-diagnosability for active on-line diagnosis in discrete event systems[J]. Automatica, 2017, 83: 220-225.

[91] Hu Y, Cao S. Asynchronous diagnosability enforcement in discrete event systems based on supervisory control[J]. IEEE Sensors Journal, 2023, 23(9): 10071-10079.

[92] Hu Y, Hu S, Li X, et al. Supervisor synthesis for asynchronous diagnosability enforcement in labeled Petri nets[J]. Information Sciences, 2024: 120907.

[93] Debouk R, Lafortune S, Teneketzis D. On an optimization problem in sensor selection[J]. Discrete Event Dynamic Systems, 2002, 12: 417-445.

[94] Thorsley D, Teneketzis D. Active acquisition of information for diagnosis and supervisory control of discrete event systems[J]. Discrete Event Dynamic Systems, 2007, 17: 531-583.

[95] Cassez F, Tripakis S. Fault diagnosis with static and dynamic observers[J]. Fundamenta Informaticae, 2008, 88(4): 497-540.

[96] Yin X, Lafortune S. A general approach for optimizing dynamic sensor activation for discrete event systems[J]. Automatica, 2019, 105: 376-383.

[97] Hu Y, Ma Z, Li Z, et al. Diagnosability enforcement in labeled Petri nets using supervisory control[J]. Automatica, 2021, 131: 109776.

[98] Cabasino M P, Lafortune S, Seatzu C. Optimal sensor selection for ensuring diagnosability in labeled Petri nets[J]. Automatica, 2013, 49(8): 2373-2383.

[99] Ran N, Giua A, Seatzu C. Enforcement of diagnosability in labeled Petri nets via optimal sensor selection[J]. IEEE Transactions on Automatic Control, 2019, 64(7): 2997-3004.

[100] Hu S, Li Z, Wisniewski R. Optimal sensor selection for diagnosability enforcement in labeled Petri nets[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2024, 54(5): 2965-2977.

[101] Hu S, Li Z. A digital twin approach for enforcing diagnosability in Petri nets[J]. IEEE Transactions on Automation Science and Engineering, 2024, 21(4): 6068-6080.

[102] Velasquez I, Le Corronc E, Pencolé Y. Active diagnosis algorithm for the localization of time failures in (max,+)-linear systems[J]. IFAC-PapersOnLine, 2022, 55(28): 276-283.

[103] Komenda J, Lahaye S, Boimond J L. Supervisory control of (max,+) automata: A behavioral approach[J]. Discrete Event Dynamic Systems, 2009, 19: 525-549.

[104] Lahaye S, Komenda J, Boimond J L. Supervisory control of (max,+) automata: Extensions towards applications[J]. International Journal of Control, 2015, 88(12): 2523-2537.

[105] Brandin B, Su R, Lin L. Supervisory control of time-interval discrete event systems[J]. IEEE Transactions on Automatic Control, 2024, 69(5): 3080-3095.

[106] Bouyer P, D'Souza D, Madhusudan P, et al. Timed control with partial observability[C]//2003 15th International Conference on Computer Aided Verification (CAV). Springer, 2003: 180-192.

[107] Miao S, Lai A, Komenda J. Always guarding you: Strong initial-and-final-state opacity of discrete-event systems[J]. Automatica, 2025, 173: 112085.

[108] Miao S, Komenda J, Masopust T, et al. Enforcement of critical observability in modular discrete-event systems[J]. IEEE Transactions on Automatic Control, 2025, conditionally accepted.

[109] Miao S, Komenda J, Ji Y. Modular enforcement of critical observability and weak opacity in discrete event systems[Z]. 2025, pending submission.

[110] Miao S, Jančar P, Komenda J, et al. Diagnosability verification for automata and petri nets: Can we do better?[J]. Automatica, 2025, under review.

[111] Miao S, Masopust T, Lai A, et al. Deciding weak prognosability for discrete-event systems[Z]. 2025, pending submission.

[112] Miao S, Lai A, Komenda J, et al. Decentralized fault diagnosis for constant-time automata[J]. IEEE Control Systems Letters, 2025, 9: 3392-3397.

[113] Miao S, Komenda J, Lai A. Active diagnosis of time-interval automata: Time perspectives[J]. IEEE Transactions on Automation Science and Engineering, 2025, 22: 11239-11249.

[114] Dima C. Real-time automata[J]. Journal of Automata, Languages and Combinatorics, 2001, 6 (1): 3-24.

[115] Marques M G, Barcelos R J, Basilio J C. The use of time-interval automata in the modeling of timed discrete event systems and its application to opacity[J]. IFAC-PapersOnLine, 2023, 56 (2): 8654-8659.

[116] Peterson J L. Petri net theory and the modeling of systems[M]. Prentice Hall, 1981.

[117] Sipser M. Introduction to the theory of computation[M]. Cengage Learning, 2012.

[118] Shu S, Lin F. Generalized detectability for discrete event systems[J]. Systems & Control Letters, 2011, 60(5): 310-317.

[119] Shu S, Lin F. I-detectability of discrete-event systems[J]. IEEE Transactions on Automation Science and Engineering, 2013, 10(1): 187-196.

[120] Hickey T, Ju Q, Van Emden M H. Interval arithmetic: From principles to implementation[J]. Journal of the ACM, 2001, 48(5): 1038-1068.

[121] Stearns R E, Hartmanis J, Lewis P M. Hierarchies of memory limited computations[C]//1965 6th Annual Symposium on Switching Circuit Theory and Logical Design (SWCT). IEEE, 1965: 179-190.

[122] Zhang K, Giua A. k-delayed strong detectability of discrete-event systems[C]//2019 58th IEEE Conference on Decision and Control (CDC). IEEE, 2019: 7647-7652.

[123] Zhang K. A unified method to decentralized state detection and fault diagnosis/prediction of discrete-event systems[J]. Fundamenta Informaticae, 2021, 181(4): 339-371.

[124] Zhang K. A new framework for discrete-event systems[J]. Foundations and Trends® in Systems and Control, 2023, 10(1-2): 1-179.

[125] Zhang K. A unified concurrent-composition method to state/event inference and concealment

in labeled finite-state automata as discrete-event systems[J]. Annual Reviews in Control, 2023, 56: 100902.

[126] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to algorithms[M]. MIT Press, 2022.

[127] Schmidt K, Breindl C. Maximally permissive hierarchical control of decentralized discrete event systems[J]. IEEE Transactions on Automatic Control, 2010, 56(4): 723-737.

[128] Feng L, Wonham W M. On the computation of natural observers in discrete-event systems[J]. Discrete Event Dynamic Systems, 2010, 20: 63-102.

[129] Pena P N, Cury J E R, Lafortune S. Verification of nonconflict of supervisors using abstractions [J]. IEEE Transactions on Automatic Control, 2009, 54(12): 2803-2815.

[130] Schmidt K, Moor T, Perk S. Nonblocking hierarchical control of decentralized discrete event systems[J]. IEEE Transactions on Automatic Control, 2008, 53(10): 2252-2265.

[131] Komenda J, Masopust T, van Schuppen J H. Coordination control of discrete-event systems revisited[J]. Discrete Event Dynamic Systems, 2015, 25: 65-94.

[132] Komenda J, Masopust T, van Schuppen J H. On conditional decomposability[J]. Systems & Control Letters, 2012, 61(12): 1260-1268.

[133] Thorsley D, Teneketzis D. Diagnosability of stochastic discrete-event systems[J]. IEEE Transactions on Automatic Control, 2005, 50(4): 476-492.

[134] Bertrand N, Haddad S, Lefaucheux E. Foundation of diagnosis and predictability in probabilistic systems[C]//2014 34th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). 2014: 417-429.

[135] Chen J, Keroglou C, Hadjicostis C N, et al. Revised test for stochastic diagnosability of discrete-event systems[J]. IEEE Transactions on Automation Science and Engineering, 2016, 15(1): 404-408.

[136] Meyer A R, Stockmeyer L J. The equivalence problem for regular expressions with squaring requires exponential space[C]//1972 13th Annual Symposium on Switching and Automata Theory (SWAT). 1972: 125-129.

[137] Krötzsch M, Masopust T, Thomazo M. Complexity of universality and related problems for partially ordered NFAs[J]. Information and Computation, 2017, 255: 177-192.

[138] Czerwiński W, Orlikowski Ł. Reachability in vector addition systems is Ackermann-complete [C]//2021 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2021: 1229-1240.

[139] Leroux J. The reachability problem for Petri nets is not primitive recursive[C]//2021 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2021: 1241-1252.

[140] Hack M H T. Decidability questions for Petri nets[D]. Massachusetts Institute of Technology, 1976.

[141] Tarjan R. Depth-first search and linear graph algorithms[J]. SIAM Journal on Computing, 1972, 1(2): 146-160.

[142] Allauzen C, Mohri M, Rastogi A. General algorithms for testing the ambiguity of finite au-

tomata and the double-tape ambiguity of finite-state transducers[J]. International Journal of Foundations of Computer Science, 2011, 22(04): 883-904.

[143] Brailsford S C, Potts C N, Smith B M. Constraint satisfaction problems: Algorithms and applications[J]. European Journal of Operational Research, 1999, 119(3): 557-581.

[144] Masopust T, Osička P. On algorithms verifying initial-and-final-state opacity: Complexity, special cases, and comparison[J]. Automatica, 2025, 174: 112171.

[145] Balun J, Masopust T, Osička P. Speed me up if you can: Conditional lower bounds on opacity verification[C]//2023 48th International Symposium on Mathematical Foundations of Computer Science (MFCS). 2023.

[146] Mohajerani S, Ji Y, Lafortune S. Compositional and abstraction-based approach for synthesis of edit functions for opacity enforcement[J]. IEEE Transactions on Automatic Control, 2019, 65(8): 3349-3364.

[147] Liu Y, Komenda J, Li Z. Supervisory control of multiagent discrete-event systems with partial observation[J]. IEEE Control Systems Letters, 2021, 6: 1867-1872.

[148] Lin F. Control of networked discrete event systems: Dealing with communication delays and losses[J]. SIAM Journal on Control and Optimization, 2014, 52(2): 1276-1298.

[149] Luo Y, Miao S, Yu X, et al. Supervisory control of weighted automata with control delays[J]. IEEE Control Systems Letters, 2025, under review.

[150] Zheng S, Shu S, Lin F. Modeling and control of discrete event systems under joint sensor-actuator cyber attacks[J]. IEEE Transactions on Control of Network Systems, 2024, 11(2): 782-794.

[151] Luo Y, Miao S, Lai A. State estimation for discrete-event systems with reliable states under sequential attacks[J]. European Journal of Control, 2025, under review.

[152] Miao S, Komenda J, Lin F. Hierarchical supervisory control of networked and cyber-attacked discrete-event systems[J]. Automatica, 2025, under review.

[153] Yin X, Lafortune S. Codiagnosability and coobservability under dynamic observations: Transformation and verification[J]. Automatica, 2015, 61: 241-252.

[154] Ji Y, Yin X, Xiao W. Supervisory control for stabilization under multiple local average payoff constraints[C]//2021 60th IEEE Conference on Decision and Control (CDC). IEEE, 2021: 1054-1061.

[155] Mertin N F A, Rudie K. Generalizing discrete-event system control problems to optimal control [C]//2024 American Control Conference (ACC). IEEE, 2024: 5433-5440.

[156] Lefebvre D, Li Z, Liang Y. Diagnosis of timed patterns for discrete event systems by means of state isolation[J]. Automatica, 2023, 153: 111045.

[157] Ma Z, Tong Y, Seatzu C. Verification of pattern-pattern diagnosability in partially observed discrete event systems[J]. IEEE Transactions on Automatic Control, 2023.