IMITATOR4AMAS：面向 STCTL 的策略综合

Davide Catta LIPN，CNRS
UMR 7030，Université Sorbonne Paris Nord 维勒塔讷兹，法国
davide.catta@lipn.univ-paris13.fr

Adrien Lacroix LIPN，CNRS
UMR 7030，Université Sorbonne Paris Nord 与 Ecole Centrale de Nantes，法国
Adrien.Lacroix@eleves.ec-nantes.fr

Wojciech Penczek 计算机科学研究所，波兰科学院华沙，波兰
penczek@ipipan.waw.pl

Laure Petrucci LIPN，CNRS
UMR 7030，Université Sorbonne Paris Nord 维勒塔讷兹，法国
laure.petrucci@lipn.univ-paris13.fr

Teofil Sidoruk
波兰科学院计算机科学研究所，波兰华沙
t.sidoruk@ipipan.waw.pl

arXiv:2602.10810v1 [cs.LO] 11 Feb 2026

摘要
IMITATOR4AMAS 支持对 STCTL 的模型检测，以及对不完美信息下的无记忆策略进行综合。该逻辑在以异步方式执行的参数化定时自动机网络上进行解释。作为对验证器 IMITATOR 的扩展，IMITATOR4AMAS 是首个用于这一场景下策略综合的工具。实验结果表明，与以往方法相比，该工具显著提升了速度。

关键词
多智能体系统，实时，策略能力，模型检测

1 引言

多智能体系统将计算系统视为理性智能体之间交互的结果，这些智能体可以独立行动或合作以实现目标 [35]。因此，研究多智能体系统意味着考察一组智能体的策略能力 [14, 26]。为了形式化验证其交互所产生的执行是否符合预期，人们引入了策略逻辑，通过策略算子扩展时序逻辑 [2, 7, 10, 14, 15]。策略逻辑能够表达一组智能体拥有一种策略，可以确保某个给定的时序性质成立，而不受该组之外的智能体所执行行动的影响。使用策略逻辑进行形式化验证涉及两个主要问题 [11] ，如下所述。给定系统模型和一个策略公式：

(1) **模型检测：** 确定该公式在模型的初始状态下是否成立。
(2) **策略综合：** 显式构造一种策略，确保满足该时序性质。

解决后一个问题自然也就解决了前一个问题。

战略定时计算树逻辑（STCTL） [7] 是一种强大的逻辑，旨在描述执行过程受明确时间约束的系统行为。它为刻画这类情境提供了自然的框架：动作的时机并非无关紧要，而是在决定系统能否满足预期规范方面发挥着决定性作用。本文聚焦于为对系统信息不完美的智能体综合无记忆策略。这一假设至关重要：在许多现实场景中，智能体无法完全观察全局状态，其决策只能依赖于对系统的观察。因此，在不完美信息条件下开展研究，能够更真实地反映战略推理 [10]。

应用领域。 在工业自动化、航空电子和分布式控制系统等安全关键且时间敏感的多智能体应用中，策略综合尤为重要，因为正确性和及时执行至关重要 [30, 32, 34]。表示相关安全场景的攻击-防御树 [20] 可以轻松转换为本工具的多智能体环境 [9]，并在对立联盟这一全新的战略背景下进行研究 [6, 28]。社会技术交互（例如电子投票 [22] ）也由此成为一种应用场景。

2 工具比较

对多智能体系统中的策略进行推理，催生了多种验证工具，其中大多数侧重于模型检测。值得注意的示例包括 MCMAS [25] 和 MCMAS-SLK [12, 13], ，它们支持无定时战略逻辑。类似地，MOCHA [3, 33] 和 STV [21, 23, 24]面向战略验证，也支持不完美信息，但缺少时间约束。近期方法则将可满足性模理论应用于战略逻辑。MsATL [27] 通过 SMT 检查 ATL 的可满足性；SGSAT 支持SL[SG] [17]；而 SMT4SMTL [18] 则支持对 SMTL [18], 这一基于线性时序逻辑的定时战略逻辑进行模型检测 [29]。最初的STCTL 模型检测方法在 IMITATOR [4], 中使用参数 [7]对策略进行编码，但这种做法远非最优。基于重写逻辑的 Maude [8], 可支持完整 STCTL 的模型检测和策略综合STCTL。它并非专用模型检查器，因此用户需要具备专业知识来编写规格，并最优地引导深度优先搜索或 BFS 探索（IMITATOR 仅使用后者）。IMITATOR4AMAS 原生支持用于模型检测和所有策略综合的策略能力。它在这些工作的基础上进一步发展，全面支持 STCTL，并且速度远超 IMITATOR 和 Maude。

3 理论背景

本节将介绍相关的理论背景。

多智能体系统。 连续时间异步多智能体系统（CAMAS） [7]将智能体建模为定时自动机网络的组成部分。时钟约束通过转移守卫和状态不变式 [1]加以约束。智能体各自独立地交错执行私有动作，但必须在共享动作上同步。它们的乘积（或 CAMAS模型）刻画了系统的全局行为。

策略能力。 用于定义（若干）智能体所作选择的条件计划称为（联合）策略，并根据智能体可用的状态信息（完美信息 I 或不完美信息 i）和记忆（完美回忆 R 或无记忆 r）进行分类 [31]。我们关注不完全信息无记忆策略（ir-strategies），即仅基于当前局部状态作出决策的策略。

逻辑。 战略定时 CTL （STCTL） [7] 通过添加时间约束和战略算子扩展了 CTL 。其语法定义如下：

$$\varphi ::= \mathrm{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle\gamma,$$
$$\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \forall\gamma \mathrm{U}_I \gamma \mid \exists\gamma \mathrm{U}_I \gamma \mid \forall\gamma \mathrm{R}_I \gamma \mid \exists\gamma \mathrm{R}_I \gamma,$$

其中，p 是一个原子命题， A 是智能体的一个子集，$A\gamma$表示联盟 A 具有实施性质γ的策略，而区间 $I \subseteq \mathbb{R}_{0+}$ 表示对时序算子求值的时间约束。其余的 CTL 路径量词（∀、∃）、时序算子（U_I、R_I，其中 F_I、G_I为其派生算子）以及布尔连接词（¬、∧）均为标准定义（其形式语义见 [7]）。

4 策略综合的架构、技术与应用

IMITATOR4AMAS 的实现扩展了现有的模型检查器 IMITATOR [4]，该工具以 OCaml 编写。IMITATOR 是一款先进的工具，支持对以定时自动机（TA）网络表示的实时系统中的**定时计算树逻辑**属性进行验证，并支持异步执行。值得注意的是，它还支持参数化验证，即输入模型为参数化定时自动机，其中表示未知值常量的参数可以出现在转移守卫和局部状态不变式中。因此，通过广度优先搜索（BFS）构造的全局状态空间节点，是由自动机位置以及关于参数和时钟的约束组成的元组。

在策略语境中，输入模型中的自动机被视为自主的智能体，它们具有各自的行为，并通过共享动作进行同步。它们执行的转移带有动作标签，这些动作是策略的一部分。为了使 IMITATOR 适用于STCTL 模型检测和策略综合，全局状态中加入了对当前正在构造的策略的表示。在 BFS 探索过程中，只有当后继状态的策略与源状态已选定的策略兼容，或源状态尚无策略（此时会为目标状态更新策略）时，才能将该后继状态加入状态空间。因此，构造出的状态空间仅包含与该策略一致的执行（称为该策略的结果路径）。在运行过程中检查满足策略算子内**定时计算树逻辑**属性的状态，工具会显示有效策略及其关联参数[1]。请注意，此过程可能无法终止，但得到的部分结果是可靠的。

IMITATOR4AMAS 代码库和配套制品可通过 https://hub.docker.com/r/imitator/aamas2026 获取，工具的演示视频请访问 https://youtu.be/E1VpFxotNG0。

5 实验评估

我们在投票示例 [7], 的基准测试中，将 IMITATOR4AMAS 与先前采用 IMITATOR [7], 的方法进行效率比较，同时还扩大了联盟规模。公式 $V_i \exists \mathrm{F}_{[0;8]}$投票$_{i,1}$ 指定投票者 V_i 有一个策略，能在 8 个时间单位内投给第一位候选人。在实验中，我们综合所有 策略（通过输入带有 synth 关键字的性质）。结果汇总于表 1，显示与 [7] 的初始方法相比有显著提升（最高达 35 倍）。所有时间均以秒为单位，超时时间设为 120 秒。

		IMITATOR				IMITATOR4AMAS			
|A|	v	c=1	c=2	c=3	c=4	c=1	c=2	c=3	c=4
1	1	0.01	0.01	0.02	0.03	0.01	0.01	0.01	0.01
1	2	0.06	0.14	0.38	0.70	0.02	0.03	0.04	0.05
1	3	0.67	2.15	6.49	16.5	0.10	0.21	0.44	0.81
1	4	6.03	26.4	78.0	超时时间	0.76	2.74	7.95	21.3
1	5	45.4	超时时间			9.51	90.8	超时时间	
2	2	0.09	0.25	0.62	1.38	0.02	0.03	0.04	0.05
2	3	0.92	3.55	10.0	27.5	0.12	0.27	0.53	1.02
2	4	10.4	53.9	超时时间		1.01	3.75	12.0	34.0
2	5	98.1	超时时间			15.5	超时时间		
3	3	1.21	5.16	17.2	47.2	0.12	0.29	0.65	1.33
3	4	14.4	67.3	超时时间		1.18	4.72	16.4	53.7
3	5	超时时间				20.3	超时时间		

表 1：综合投票基准测试中所有策略，包含 v 名选民、 c 名候选人，以及规模为 $|A|$ 的智能体联盟。

此外，演示视频还展示了其他示例：添加了时钟的寻宝者 [28] ，将其指定为参数化定时自动机，以展示定时约束综合；以及 [16]的会议场景变体。我们请读者参阅 [8, 表 1]，其中列出了来自库 [5], 的一系列参数化定时自动机基准测试，在许多实例中，IMITATOR（因此 IMITATOR4AMAS 也一样）的表现优于 Maude。尽管这些基准测试涉及非策略属性，但我们注意到，IMITATOR4AMAS 能够对相同模型进行验证，并支持涉及策略能力的表达力更强的规约。

6 结论与未来工作

IMITATOR4AMAS 提供了一个集成环境，可通过一键式流程综合不完美信息下的无记忆策略。用户只需提供模型和属性。与其他方法相比，它不需要用户具备额外的专业知识。该工具是实时多智能体系统策略综合领域向前迈出的重要一步。

进一步扩展 IMITATOR4AMAS 的一个重要方向，是支持其他策略语义，例如计数 [19],定时，或在不完美信息与完全信息之间引入一种中间的“部分视图”，使策略选择基于部分智能体的局部状态（这些智能体不一定属于该联盟）。

致谢

本研究得到以下项目的支持：CNRS IRP “Le Trójkąt”、ANR-22-CE48-0012 项目 BISOUS，以及 PHC Polonium 项目 MoCcA（BPN/BFR/2023/1/00045）。我们感谢 Jaime Arias 为本文的演示视频和配套制品提供帮助。

参考文献

[1] Rajeev Alur 和 David Dill。1992。《定时自动机理论》。载于《实时：理论与实践》。施普林格，45–73。 [2] Rajeev Alur、Thomas A. Henzinger 和 Orna Kupferman。2002。交替时间时序逻辑。《*ACM*学报》49(5)（2002），672–713。 [3] Rajeev Alur、Thomas. A. Henzinger、Freddy Y. C. Mang、Shaz Qadeer、Sriram K. Rajamani 和 Serdar Tasiran。1998。MOCHA：模型检验中的模块化。载于*CAV 1998*论文集。施普林格，521–525。

[1]通过扩展 IMITATOR，即可免费获得时序参数综合结果。

[4] Étienne André。2021。IMITATOR 3：超越可判定性的时序参数综合。载于*CAV 2021*论文集。施普林格，552–565。[5] Étienne André、Dylan Marinho 和 Jaco van de Pol。2021。扩展参数化定时自动机基准测试库。载于*TAP 2021*论文集。施普林格，39–50。[6] Jaime Arias、Carlos Budde、Wojciech Penczek、Laure Petrucci、Teofil Sidoruk 和 Mariëlle Stoelinga。2020。黑客与安全：作为异步多智能体系统的攻击-防御树。载于*ICFEM 2020*论文集。施普林格，3–19。[7] Jaime Arias、Wojciech Jamroga、Wojciech Penczek、Laure Petrucci 和 Teofil Sidoruk。2023。战略（定时）计算树逻辑。载于*AAMAS'23*论文集。ACM，382–390。[8] Jaime Arias、Carlos Olarte、Wojciech Penczek、Laure Petrucci 和 Teofil Sidoruk。2024。使用重写逻辑中的策略对战略定时 CTL 进行模型检测与综合。载于*PPDP 2024*论文集。ACM，10:1–10:14。[9] Jaime Arias、Wojciech Penczek、Laure Petrucci 和 Teofil Sidoruk。2021。ADT2AMAS：管理攻击-防御场景中的智能体。载于*AAMAS '21*论文集。ACM，1749–1751。[10] Raphaël Berthon、Bastien Maubert、Aniello Murano、Sasha Rubin 和 Moshe Y. Vardi。2017。不完美信息策略逻辑。载于*LICS 2017*论文集。IEEE计算机学会，1–12。[11] N. Bulling、J. Dix 和 W. Jamroga。2010。战略能力逻辑的模型检测：复杂性。载于多智能体系统的规范与验证。施普林格，125–159。[12] Petr Cermák、Alessio Lomuscio、Fabio Mogavero 和 Aniello Murano。2014。MCMAS-SLK：用于验证策略逻辑规范的模型检查器。载于*CAV 2014*论文集。施普林格，525–532。[13] Petr Cermák、Alessio Lomuscio、Fabio Mogavero 和 Aniello Murano。2018。针对 SLK 规范的多智能体系统实用验证。*Inf. Comput.*261（2018），588–614。[14] Wojciech Jamroga。2015。多智能体系统规范与验证的逻辑方法。ICS PAS出版社。[15] Wojciech Jamroga。2018。战略能力的模型检测——为何、检测什么，以及尤其是如何检测？载于*TIME 2018*论文集。Schloss Dagstuhl - Leibniz信息学中心，1–10。[16] Wojciech Jamroga、Wojciech Penczek 和 Teofil Sidoruk。2021。异步智能体的战略能力：语义副作用及其驯服之道。载于*KR 2021*论文集。368–378。[17] Magdalena Kacprzak 、 Artur Niewiadomski 和 Wojciech Penczek。2021。具有简单目标的策略逻辑可满足性检查。载于*KR 2021*论文集。400–410。[18] Magdalena Kacprzak、Artur Niewiadomski、Wojciech Penczek 和 Andrzej Zbrzezny。2023。战略度量时序逻辑的基于SMT的可满足性检查。载于*ECAI 2023*论文集。IOS Press，1180–1189。[19] Michal Knapik、Étienne André、Laure Petrucci、Wojciech Jamroga 和 Wojciech Penczek。2019。定时ATL：忘掉记忆，只需计数。*J. Artif. Intell. Res.* 66（2019），197–223。[20] Barbara Kordy、Sjouke Mauw、Saˇsa Radomirović 和 Patric Schweitzer。2011。《攻击-防御树基础》。载于*FAST 2010* 论文集。施普林格，80–95。[21] Damian Kurpiewski、Wojciech J amroga 和 Michal Knapik。2019。STV：不完美信息下策略的模型检测。载于*AA- MAS'19* 论文集。2372–2374。[22] Damian Kurpiewski、Wojciech Jamroga、Łukasz Maśko、Łukasz Mikulski、Witold Pazderski、Wojciech Penczek 和 Teofil Sidoruk。2022。电子投票中多智能体属性的验证：案例研究。载于*AiML 2022* 论文集。College Publications，531–556。[23] Damian Kurpiewski、Mateusz Kaminski 和 Wojciech Jamroga。2024。STV+FLY：多智能体系统中战略能力的在线模型检测。载于*ECAI 2024* 论文集。IOS Press，4483–4486。[24] Damian Kurpiewski、Witold Pazderski、Wojciech J amroga 和 Yan Kim。2021。STV+约简：使用模型约简实现战略能力的实用验证。载于*AAMAS'21* 论文集。ACM，1770–1772。[25] Alessio Lomuscio、Hongyang Qu 和 Franco Raimondi。2015。MCMAS：用于验证多智能体系统的开源模型检测器。《国际软件工具与技术转移杂志》24（2015），84–90。[26] Alessio Lomuscio 和 Franco Raimondi。2006。在多智能体系统中对知识、策略与博弈进行模型检测。载于自主智能体与多智能体系统国际会议论文集（*AAMAS' 06*）。161–168。[27] Artur Niewiadomski、Magdalena Kacprzak、Damian Kurpiewski、Michal Knapik、Wojciech Penczek 和 Wojciech Jamroga。2020。MsATL：一种基于 SAT 的 ATL 可满足性检查工具。载于*AAMAS'20* 论文集。IFAAMAS，2111–2113。[28] Laure Petrucci、Michal Knapik、Wojciech Penczek 和 Teofil Sidoruk。2019。压缩（攻击-防御）树的状态空间。载于*ICECCS 2019* 论文集。IEEE，71–80。[29] Amir Pnueli。1977。程序的时序逻辑。载于*SFCS '77* 论文集。IEEE计算机学会，46–57。[30] Christine Rochange。2016。从最坏执行时间分析和任务调度方法的角度看并行实时任务。载于*WCET 2016* 论文集。Schloss Dagstuhl - Leibniz信息学中心，1–11。[31] Pierre-Yves Schobbens。2004。不完美回忆交替时间逻辑。载于*LCMAS 2003* 论文集。爱思唯尔，1–12。[32] Abhishek Singh。2024。可抢占单处理器调度问题的割平面算法。《实时系统》60，1（2024），24–73。[33] Wiebe van der Hoek 和 Michael Wooldridge。2002。认知目标的可处理多智能体规划。载于*AAMAS'02* 论文集。ACM，1167–1174。[34] Marilyn Wolf。2023。第9章——汽车与航空航天系统。载于《作为组件的计算机（第五版）》。摩根·考夫曼，437–452。[35] Michael J. Wooldridge。2009。《多智能体系统导论，第二版》。威立。

技术要求

要复现实验，请安装最新的 Docker 软件包并运行以下命令（已在通过 WSL 运行的 Ubuntu 22.04 和 macOS 上测试）。首次调用 `imitator/aamas2026` 容器时，系统会自动下载该容器。

选民示例：docker run imitator/aamas2026 /imitator/examples/voters.imi /imitator/examples/voters.imiprop -no-var-autoremove

寻宝者示例：

docker run imitator/aamas2026 /imitator/examples/treasure_clocks.imi /imitator/examples/treasure.imiprop -no-var-autoremove

会议示例：

docker run imitator/aamas2026 /imitator/examples/conf2.imi /imitator/examples/conf2.imiprop -no-var-autoremove

Linux 环境：

要访问该工具和示例的源文件，您可以运行 Docker，以便在 Linux 环境中工作：docker run –rm -it --entrypoint "/bin/bash" -v YOUR_LOCAL_EXAMPLES_DIRECTORY:/COPY_IN_THE_IMAGE -w /imitator imitator/aamas2026

其中使用了以下选项：

`-rm`：退出时清理容器文件`-it`：通过终端窗口进行交互式使用`–entrypoint "/bin/bash"`：启动容器时运行命令行解释器，而不是运行 IMITATOR 工具`-v YOUR_LOCAL_EXAMPLES_DIRECTORY:/COPY_IN_THE_IMAGE`：如果您想创建自己的示例，可以将自己机器上的文件夹链接到正在运行的 Docker 镜像中的一个文件夹。这样，您就可以像往常一样在自己的机器上轻松编辑这些文件，并在 Docker 镜像中对其进行测试。然后，将 `YOUR_LOCAL_EXAMPLES_DIRECTORY` 替换为您的文件夹路径，并将 `COPY_IN_THE_IMAGE` 替换为 Docker 中的名称。它将位于目录树的根目录下（因为使用了 /）。如果您不打算使用此功能，只需从命令行中删除该选项即可。`-w /imitator`：启动时使用的工作目录。如果您使用了前一个选项，可以写成 `-w /COPY_IN_THE_IMAGE` ，以便直接在自己的文件所在的文件夹中工作。

然后按照视频中的演示运行该工具，例如：/imitator/bin/imitator /imitator/examples/treasure_clocks.imi /imitator/examples/treasure.imiprop -no-var-autoremove