E-voting, Card Games, and Drone Teams: Verification of Strategic Properties in Multi-Agent Systems with Imperfect Information.

Phd dissertation

Damian Kurpiewski

Institute of Computer Science

Polish Academy of Sciences

Supervisor:
prof. dr hab. **Wojciech Jamroga**
Institute of Computer Science, PAS

2025 December

Abstract

The formal verification of strategic abilities in multi-agent systems (MAS) is a critical yet notoriously challenging endeavor, particularly under conditions of imperfect information and imperfect recall. The inherent computational complexity, often characterized by state-space explosion and high decision-theoretic complexity, renders the exact model checking of such systems intractable for many real-world applications. This thesis addresses these fundamental challenges by developing, implementing, and empirically evaluating a suite of novel methods for the scalable verification of strategic properties in complex, informationally restricted environments.

Our primary contribution is a comprehensive framework for approximate verification of strategic ability, formalized using Alternating-time Temporal Logic with imperfect information and memoryless strategies ($\mathbf{ATL}_{\mathrm{ir}}$). We introduce a novel fixpoint approximation scheme that provides sound lower and upper bounds for the truth value of $\mathbf{ATL}_{\mathrm{ir}}$ formulae. This is achieved by translating $\mathbf{ATL}_{\mathrm{ir}}$ specifications into an alternating epistemic μ-calculus variant, enhanced with a steadfast next-step operator that captures agents' commitment to strategies within their common knowledge neighborhoods. To manage the combinatorial complexity of epistemic relations, we devise optimized data structures based on disjoint-set forests and propose dynamic state-space reduction techniques, including model abstraction and partial-order reduction, which can be synergistically combined with our fixpoint methods. For systems exhibiting asynchronous interaction, we introduce a local model approximation technique. By constructing agent-centric abstractions of the global system, this method enables verification of strategic abilities with a substantial reduction in the state space, while preserving formal soundness guarantees.

Furthermore, we present a forward-chaining strategy synthesis algorithm based on depth-first search, enhanced with a novel dominance relation for

partial strategies. This relation enables the systematic pruning of suboptimal choices, significantly improving the search for winning strategies. We extend this approach to multi-criteria strategy optimization, allowing for the synthesis of strategies that are simultaneously effective with respect to multiple objectives, such as outcome containment and action uniformity.

To validate the practical efficacy of our theoretical contributions, we developed the StraTegic Verifier (STV), a state-of-the-art model checker that implements the proposed algorithms. We applied STV and our methods to a diverse range of challenging real-world models, including the bridge card game, autonomous drone teams, smart production factories, social explainable AI (SAI) ecosystems, and sophisticated e-voting protocols like Selene. Our experimental results demonstrate that the proposed approximation and synthesis methods yield dramatic performance improvements, often by several orders of magnitude, over existing state-of-the-art tools. In numerous instances, our approach successfully verified models with millions of states, a feat unattainable by exact methods, while maintaining a high degree of accuracy and providing conclusive results.

In conclusion, this thesis establishes a robust and scalable methodology for the verification of strategic abilities under imperfect information. By bridging the gap between theoretical logic and practical implementation, our work enables the formal analysis of complex multi-agent systems that were previously beyond the reach of automated verification, thereby advancing both the theory and practice of ensuring correctness and security in increasingly autonomous and interactive computing environments.

Streszczenie

Formalna weryfikacja zdolności strategicznych w systemach wieloagentowych (MAS) jest zadaniem krytycznym, ale niezwykle trudnym, szczególnie w warunkach niedoskonałych informacji i niedoskonałej pamięci. Nieodłączna złożoność obliczeniowa, często charakteryzująca się eksplozją przestrzeni stanów i wysoką złożonością teoretyczną decyzji, sprawia, że dokładne sprawdzanie modeli takich systemów jest niemożliwe w wielu rzeczywistych zastosowaniach. W niniejszej pracy podjęto te fundamentalne wyzwania poprzez opracowanie, wdrożenie i empiryczną ocenę zestawu nowatorskich metod skalowalnej weryfikacji właściwości strategicznych w złożonych środowiskach o ograniczonej dostępności informacji.

Naszym głównym wkładem jest kompleksowa struktura służąca do przybliżonej weryfikacji zdolności strategicznych, sformalizowana przy użyciu logiki temporalnej czasu alternującego z niepełną informacją i strategiami bez pamięci ($\mathbf{ATL}_{\mathrm{ir}}$). Wprowadzamy nowatorski schemat aproksymacji stałopunktowej, który zapewnia solidne dolne i górne granice wartości logicznej formuł $\mathbf{ATL}_{\mathrm{ir}}$. Osiąga się to poprzez przetłumaczenie specyfikacji $\mathbf{ATL}_{\mathrm{ir}}$ na zmienną wersję epistemicznego μ-rachunku, wzbogaconą o operator stałego następnego kroku, który rejestruje zaangażowanie agentów w strategie w ramach ich wspólnej wiedzy. Aby zarządzać kombinatoryczną złożonością relacji epistemicznych, opracowujemy zoptymalizowane struktury danych oparte na lasach rozłącznych zbiorów i proponujemy dynamiczne techniki redukcji przestrzeni stanów, w tym abstrakcję modelu i redukcje częściowo-porządkowe, które można synergicznie połączyć z naszymi metodami punktów stałych. W przypadku systemów wykazujących interakcje asynchroniczne wprowadzamy technikę lokalnego przybliżenia modelu. Dzięki skonstruowaniu abstrakcji globalnego systemu skoncentrowanych na agentach metoda ta umożliwia weryfikację zdolności strategicznych przy

znacznej redukcji przestrzeni stanów, zachowując jednocześnie formalne gwarancje poprawności.

Ponadto przedstawiamy algorytm syntezy strategii oparty na wyszukiwaniu w głąb, wzbogacony o nowatorską relację dominacji dla strategii częściowych. Relacja ta umożliwia systematyczne eliminowanie suboptymalnych wyborów, znacznie usprawniając wyszukiwanie strategii prowadzących do zwycięstwa. Rozszerzamy to podejście na wielokryterialną optymalizację strategii, umożliwiając syntezę strategii, które są jednocześnie skuteczne w odniesieniu do wielu celów, takich jak ograniczenie wyników i jednolitość działań.

Aby zweryfikować praktyczną skuteczność naszych teoretycznych osiągnięć, opracowaliśmy StraTegic Verifier (STV), najnowocześniejszy weryfikator modelowy, który implementuje proponowane algorytmy. Zastosowaliśmy STV i nasze metody do szerokiej gamy trudnych modeli rzeczywistych, w tym gry karcianej brydż, autonomicznych zespołów dronów, inteligentnych fabryk produkcyjnych, ekosystemów społecznie wyjaśnialnych sztucznych inteligencji (SAI) oraz zaawansowanych protokołów głosowania elektronicznego, takich jak Selene. Wyniki naszych eksperymentów pokazują, że proponowane metody aproksymacji i syntezy zapewniają znaczną poprawę wydajności, często o kilka rzędów wielkości, w porównaniu z istniejącymi najnowocześniejszymi narzędziami. W wielu przypadkach nasze podejście pozwoliło z powodzeniem zweryfikować modele zawierające miliony stanów, co jest niemożliwe do osiągnięcia przy użyciu metod dokładnych, przy zachowaniu wysokiego stopnia dokładności i zapewnieniu jednoznacznych wyników.

Podsumowując, niniejsza praca ustanawia solidną i skalowalną metodologię weryfikacji zdolności strategicznych w warunkach niedoskonałej informacji. Wypełniając lukę między logiką teoretyczną a praktyczną implementacją, nasza praca umożliwia formalną analizę złożonych systemów wieloagentowych, które wcześniej były poza zasięgiem automatycznej weryfikacji, przyczyniając się tym samym do rozwoju zarówno teorii, jak i praktyki zapewniania poprawności i bezpieczeństwa w coraz bardziej autonomicznych i interaktywnych środowiskach komputerowych.

Acknowledgements

The path to completing this dissertation has been fraught with both professional challenges and personal adversities. I am profoundly grateful for the enduring support of my supervisor, Prof. Wojciech Jamroga, whose steadfast belief in my capabilities and academic potential encouraged me to persevere even during my lowest moments. His guidance was a beacon of light in times of doubt.

I must also express my deepest appreciation for my family, whose unwavering support and understanding provided me with the strength to continue. Their patience and encouragement were invaluable throughout this journey.

I would also like to acknowledge my colleagues and friends in the academic community, whose insights and camaraderie enriched my research experience. Their constructive feedback and collaborative spirit were instrumental in shaping the quality of this work.

Finally, I extend my gratitude to the funding bodies and institutions that supported my research endeavors, enabling me to focus on my studies without financial burdens.

Contents

List of Figures

LIST OF FIGURES

List of Tables

1

Introduction

More than ever, our daily lives are intertwined with smart systems seamlessly integrated into our environments. Regular interactions with our smartphones have become a staple of our daily activities. Modern automobiles are outfitted with 30-100 microprocessors, running up to 100 million lines of software code. Smart production facilities are poised to boost goods production by up to 25 percent, potentially generating up to 1.8 trillion in new value annually across global factories. Given the profound societal impact of these embedded systems, their verification has emerged as a critical task.

Traditionally, this verification is conducted using model checking techniques: both the system and its environment are modeled as state machines, and these models are then scrutinized to determine whether they satisfy specific safety or liveness properties within a given environment. This approach is not only applicable to physical systems but also extends to the realm of digital security and integrity, particularly in verifying e-voting protocols.

E-voting systems, integral to modern democracies, necessitate rigorous verification to ensure their reliability and trustworthiness. These systems are complex amalgamations of software, hardware, and human interaction, making their verification challenging yet imperative. The verification process must encompass various crucial procedures, including but not limited to:

1. **Security Protocols Verification**: Ensuring that the e-voting system is immune to unauthorized access, manipulation, and ensures voter anonymity.

2. **Accuracy and Integrity Checks**: Verifying that every vote is accurately recorded and tallied without alteration or loss.

3. **Auditability**: Ensuring that the system maintains robust logs and audit trails for verification and recount purposes, if needed.

4. **Accessibility and Usability Verification**: Confirming that the system is accessible to all voters, including those with disabilities, and is user-friendly.

5. **Resilience to Attacks**: Testing the system's ability to withstand various forms of cyber attacks and faults, maintaining its functionality and integrity.

The complexity of these systems, especially considering the various layers of interaction (human, network, software, and hardware), makes the application of traditional model checking a sophisticated task. It demands not only a deep understanding of the theoretical underpinnings of model checking but also an appreciation of the practical aspects and specific challenges inherent in the verification of e-voting protocols.

In summary, the verification of such important systems and protocols represents a vital component in ensuring the safety, security, and efficacy of technologies that are increasingly central to our daily lives and democratic processes.

1.1 Practical Aspects of Model Checking

Model checking has emerged as a pivotal tool in the verification of computer systems, offering a systematic methodology to ascertain whether a system's model satisfies specified requirements. The verification of sociotechnical systems—integrating both technical components and human agents—introduces additional layers of complexity. These systems, marked by their dynamic behavior, multiple interactive agents, and imperfect information, demand nuanced verification approaches. Beyond theoretical interest, ensuring specific behaviors or outcomes in these systems carries substantial real-world consequences, especially in domains concerning human safety, infrastructure, or significant financial interests.

The need for practical application in these fields transcends theoretical assurances. Stakeholders require confirmation that systems will perform as anticipated in real-world

situations. Moreover, the sheer scale and intricacy of sociotechnical systems often preclude exhaustive manual analysis, underscoring the necessity for practicality in model checking, which hinges on:

1. **Scalability**: Verification efficiency and adaptability are paramount as system size and complexity expand.
2. **Usability**: Tools and methods should be accessible to users without extensive formal methods training, featuring intuitive interfaces and seamless integration into existing processes.
3. **Relevance**: Model checking should yield practical insights for enhancing sociotechnical system design and operation.

1.1.1 Challenges in Practical Model Checking

Practical model checking is challenging for a number of reasons.

Creating the Models Accurately representing a sociotechnical system involves:

1. **Complex Interactions**: Detailed modeling of agent interactions under imperfect information is challenging.
2. **Evolution Over Time**: Systems change, necessitating adaptable or easily updated models.
3. **Granularity**: Balancing detail and abstraction is crucial to maintain computational feasibility while capturing essential system aspects.

Specification of the Right Requirements (Formulas) The value of model checking is tied to the precision and relevance of the specifications, involving:

1. **Expressiveness**: Formulas must accurately reflect desired properties.
2. **Ambiguity**: Natural language requirements often lack clarity. Their translation into precise specifications is both challenging and vital.
3. **Completeness**: Comprehensive coverage of relevant behaviors and scenarios is crucial to identify potential system vulnerabilities.

In summary, while the logics used in this thesis provide a robust theoretical basis for formal verification of sociotechnical systems, their practical application are fraught with challenges. Effectively addressing these challenges is key to enhancing the reliability and trustworthiness of sociotechnical systems in real-world settings.

1.2 Thesis Statement

In the realm of formal verification, the challenge of accurately modeling and analyzing systems with multiple agents, especially under conditions of imperfect information, remains a significant task. Our research specifically addresses this challenge by studying verification algorithms for Alternating-Time Temporal Logic (**ATL**) with imperfect information and memoryless strategies. We extend our investigation to the practical application of these algorithms by defining models of real-life scenarios. A key focus is the domain of electronic voting (e-voting) systems, where we model these as both synchronous and asynchronous game structures. This approach not only allows for a more comprehensive understanding of strategic behaviors in multi-agent systems but also paves the way for more effective verification of their properties under varying informational constraints.

This thesis is dedicated to advancing the field of model checking by developing and refining verification algorithms for **ATL** with imperfect information. We focus on modeling real-life scenarios, particularly in the context of e-voting systems, using both synchronous and asynchronous game structures. Our work aims to enhance the application of **ATL** in multi-agent systems, thereby providing more robust and efficient methods for verifying strategic properties in scenarios where information is incomplete or imperfect.

1.3 Published Materials

The material in this thesis is based on the following papers, to which the author has contributed:

- (Kurpiewski et al. (2025)) Damian Kurpiewski, Wojciech Jamroga, and Yan Kim. **Approximate Verification of Strategic Abilities under Imperfect Information Using Local Models**. In *Proceedings of the Thirty-Fourth*

International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025, pages 143–151. ijcai.org, 2025.
In this work, my main contribution was to the theory behind the proposed approach. I defined the concept of approximated local models and, importantly, I constructed and presented the proof that these models can effectively approximate the verification of specified properties. This theoretical foundation shows that our method is sound and offers the formal guarantees needed for its practical use in solving complex verification problems with imperfect information.

- (Jamroga and Kurpiewski (2023)) Wojciech Jamroga and Damian Kurpiewski. **Pretty Good Strategies and Where to Find Them**. In Vadim Malvone and Aniello Murano, editors, *Multi-Agent Systems - 20th European Conference, EUMAS 2023, Naples, Italy, September 14-15, 2023, Proceedings*, **14282** of *Lecture Notes in Computer Science*, pages 363–380. Springer, 2023.
My research was dedicated to the development of innovative algorithms aimed at enhancing outcomes through domination-based strategy optimization. In pursuit of validating these algorithms, I designed a novel procedure for generating randomized models. These models served as a critical testing ground, enabling a comprehensive evaluation of the algorithms' effectiveness in various scenarios.

- (Kurpiewski et al. (2023)) Damian Kurpiewski, Wojciech Jamroga, and Teofil Sidoruk. **Towards Modelling and Verification of Social Explainable AI**. In Ana Paula Rocha, Luc Steels, and H. Jaap van den Herik, editors, *Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 1, Lisbon, Portugal, February 22-24, 2023*, pages 396–403. SCITEPRESS, 2023.
In this work, I have designed various models for Social Explainable Artificial Intelligence (SAI), encompassing both honest agents and potential attacks. These models were implemented in the STV tool. Additionally, I identified and designed significant security properties for verification purposes and expressed them using $\mathbf{ATL}_{\mathrm{ir}}$ formulas. I've also prepared experiments to demonstrate the efficacy and robustness of these models and properties.

- (Jamroga et al. (2022b)) Wojtek Jamroga, Lukasz Masko, Lukasz Mikulski, Witold Pazderski, Wojciech Penczek, Teofil Sidoruk, and Damian

Kurpiewski. **Verification of Multi-Agent Properties in Electronic Voting: A Case Study**. In David Fernández-Duque, Alessandra Palmigiano, and Sophie Pinchinat, editors, *Advances in Modal Logic, AiML 2022, Rennes, France, August 22-25, 2022*, pages 531–556. College Publications, 2022.
In the scope of this research, my contributions were centered on overseeing the development of a parallelized verification algorithm, which included establishing the foundational framework necessary for the algorithm's implementation. Additionally, I was responsible for the design and implementation of enhanced versions of the case study models, aiming to refine their utility and applicability. A significant aspect of my work also involved optimizing the implementation of the model reduction method, ensuring efficiency and effectiveness in the verification process. To validate the advancements made through these efforts, I orchestrated and conducted a comprehensive set of experiments, designed to rigorously assess the performance and scalability of the developed algorithm and the improved models.

- (Jamroga et al. (2022a)) Wojciech Jamroga, Damian Kurpiewski, and Vadim Malvone. **How to measure usable security: Natural strategies in voting protocols**. *J. Comput. Secur.*, **30**(3):381–409, 2022.
  My research focused on the formal analysis of a selected electronic voting protocol, which involved the careful design of sophisticated models to represent potential interactions—both correct and incorrect—between voters and the system. These models were subsequently implemented in the Uppaal tool. Furthermore, I developed several natural strategies for voters and verified their effectiveness by directly implementing these strategies within the models and utilizing the Uppaal tool for verification.

- (Kurpiewski et al. (2021)) Damian Kurpiewski, Witold Pazderski, Wojciech Jamroga, and Yan Kim. **STV+Reductions: Towards Practical Verification of Strategic Ability Using Model Reductions**. In *Proceedings of AAMAS*, pages 1770–1772. ACM, 2021.
  For this paper, I enhanced the graphical interface of the STV tool by introducing new functionalities to demonstrate the implemented reduction methods: partial order reductions and automated bisimulation checking. Additionally, I developed

a novel model specification language and implemented a corresponding parser within the tool, facilitating the loading of model specifications from text files. Lastly, I compiled experimental results to validate the efficiency of the partial order reduction method, illustrating its effectiveness in practical scenarios.

- (Jamroga et al. (2020b)) Wojciech Jamroga, Yan Kim, Damian Kurpiewski, and Peter Y. A. Ryan. **Towards Model Checking of Voting Protocols in Uppaal**. In *Proceedings of E-Vote-ID*, **12455** of *Lecture Notes in Computer Science*, pages 129–146. Springer, 2020.
  I contributed to the design and implementation of an electronic voting model using the Uppaal verification tool. Additionally, I was involved in employing Uppaal to verify selected properties of the protocol, ensuring its reliability and security.

- (Jamroga et al. (2019a)) Wojciech Jamroga, Michał Knapik, Damian Kurpiewski, and Łukasz Mikulski. **Approximate Verification of Strategic Abilities under Imperfect Information**. *Artificial Intelligence*, **277**, 2019.
  I have developed the STV model-checker, a sophisticated tool designed for the verification of $\mathbf{ATL}_{\mathrm{ir}}$ formulas. This development was underpinned by theoretical research, from which I designed and implemented a fixpoint approximation algorithm. To demonstrate the algorithm's effectiveness, I designed and implemented various models within the tool, including the castles model and the bridge game endplay model.

  Additionally, I have significantly contributed to the enhancement of the tool's model-checking capabilities. This included the development of model abstractions, the optimization of algorithms using a disjoint-union data structure, and the optimization of model generation through the exclusion of irrelevant parts of the model already generated. My responsibilities also extended to preparing and conducting experiments to test these newly implemented methods, thereby proving their effectiveness and efficiency.

- (Kurpiewski et al. (2019b)) Damian Kurpiewski, Michał Knapik, and Wojciech Jamroga. **On Domination and Control in Strategic Ability**. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*, pages 197–205. IFAAMAS, 2019.

I have made significant contributions to the design of a novel strategy synthesis algorithm, which is based on the Depth-First Search (DFS) algorithm. This involved the development of multiple heuristics for the algorithm, which were subsequently implemented along with the new method in the STV tool. Furthermore, I have developed and implemented new models simulating scenarios where flying drones search for pollution. These models were utilized to evaluate the effectiveness of the newly developed method, demonstrating its applicability and efficiency in real-world scenarios.

- (Kurpiewski et al. (2019a)) Damian Kurpiewski, Wojciech Jamroga, and Michał Knapik. **STV: Model Checking for Strategies under Imperfect Information**. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*, pages 2372–2374. IFAAMAS, 2019.
  I developed a graphical interface for the STV model-checker, enhancing its usability and functionality. This interface enables users to generate one of the preimplemented models by specifying parameters. It visually represents the generated model as a transition graph, offering insights into various aspects of the model, including state properties and actions associated with transitions. Furthermore, the interface facilitates the verification of selected formulas, displaying the outcomes directly to the user. This development significantly improves the tool's accessibility and effectiveness, allowing for a more interactive and informative user experience.

- (Kurpiewski and Marmsoler (2019)) Damian Kurpiewski and Diego Marmsoler. **Strategic logics for collaborative embedded systems**. *SICS Softw.-Intensive Cyber Phys. Syst.*, **34**(4):201–212, 2019.
  In this research, I developed a comprehensive model to simulate collaborative interactions between robots and machines within a factory environment. This model was crafted in various iterations to encompass a range of scenarios, including variations in battery capacity and storage area capacities, to accurately reflect the diverse operational conditions that might be encountered. Furthermore, I identified and formulated several key properties that capture the essence of these interactions, representing them through $\mathbf{ATL}_{\mathrm{ir}}$ formulas for rigorous analysis.

Subsequently, these models were integrated into the STV tool to facilitate detailed examination and verification. Through this integration, I conducted a series of experiments aimed at evaluating the model's performance and the feasibility of the proposed collaborative strategies. These experiments provided valuable insights into the operational dynamics and potential efficiency improvements within robot-machine collaboration in a factory setting.

1.4 Other Publications by the Candidate

Aside from the papers mentioned above, there are other papers to which the author of the thesis has also contributed:

- (Kaminski et al. (2025)) Mateusz Kaminski, Damian Kurpiewski, and Wojciech Jamroga. **NatSTV: Towards Verification of Natural Strategic Ability**. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 11072–11076. ijcai.org, 2025.

- (Kaminski et al. (2024)) Mateusz Kaminski, Damian Kurpiewski, and Wojciech Jamroga. **STV+KH: Towards Practical Verification of Strategic Ability for Knowledge and Information Flow**. In Mehdi Dastani, Jaime Simão Sichman, Natasha Alechina, and Virginia Dignum, editors, *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024*, pages 2812–2814. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024.

- (Kurpiewski et al. (2024)) Damian Kurpiewski, Mateusz Kaminski, and Wojciech Jamroga. **STV+FLY: On-the-Fly Model Checking of Strategic Ability in Multi-Agent Systems**. In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz, editors, *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de*

Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024), **392** of *Frontiers in Artificial Intelligence and Applications*, pages 4483–4486. IOS Press, 2024.

- (Jamroga et al. (2024)) Wojciech Jamroga, Yan Kim, and Damian Kurpiewski. **Scalable Verification of Social Explainable AI by Variable Abstraction**. In Ana Paula Rocha, Luc Steels, and H. Jaap van den Herik, editors, *Proceedings of the 16th International Conference on Agents and Artificial Intelligence, ICAART 2024, Volume 1, Rome, Italy, February 24-26, 2024*, pages 149–158. SCITEPRESS, 2024.

- (Mikulski et al. (2022a)) Łukasz Mikulski, Wojciech Jamroga, and Damian Kurpiewski. **Towards Assume-Guarantee Verification of Strategic Ability**. In *Proc. of AAMAS'22*, pages 1702–1704. IFAAMAS, 2022.

- (Mikulski et al. (2022c)) Lukasz Mikulski, Wojciech Jamroga, and Damian Kurpiewski. **Towards Assume-Guarantee Verification of Strategic Ability**. In Piotr Faliszewski, Viviana Mascardi, Catherine Pelachaud, and Matthew E. Taylor, editors, *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, pages 1702–1704. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022.

- (Mikulski et al. (2022b)) Lukasz Mikulski, Wojciech Jamroga, and Damian Kurpiewski. **Assume-Guarantee Verification of Strategic Ability**. In Reyhan Aydogan, Natalia Criado, Jérôme Lang, Víctor Sánchez-Anguix, and Marc Serramia, editors, *PRIMA 2022: Principles and Practice of Multi-Agent Systems - 24th International Conference, Valencia, Spain, November 16-18, 2022, Proceedings*, **13753** of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2022.

- (Kurpiewski et al. (2022)) Damian Kurpiewski, Lukasz Mikulski, and Wojciech Jamroga. **STV+AGR: Towards Verification of Strategic Ability Using Assume-Guarantee Reasoning**. In Reyhan Aydogan, Natalia Criado, Jérôme Lang, Víctor Sánchez-Anguix, and Marc Serramia, editors, *PRIMA 2022: Principles and Practice of Multi-Agent Systems - 24th*

International Conference, Valencia, Spain, November 16-18, 2022, Proceedings, **13753** of *Lecture Notes in Computer Science*, pages 691–696. Springer, 2022.

- (Jamroga et al. (2020d)) Wojciech Jamroga, Damian Kurpiewski, and Vadim Malvone. **Natural Strategic Abilities in Voting Protocols**. In Thomas Gross and Luca Viganò, editors, *Socio-Technical Aspects in Security and Trust - 10th International Workshop, STAST 2020, Virtual Event, September 14, 2020, Revised Selected Papers*, **12812** of *Lecture Notes in Computer Science*, pages 45–62. Springer, 2020.

- (Jamroga et al. (2020c)) Wojciech Jamroga, Beata Konikowska, Wojciech Penczek, and Damian Kurpiewski. **Multi-valued Verification of Strategic Ability**. *Fundamenta Informaticae*, **175**(1-4):207–251, 2020.

- (Belardinelli et al. (2019)) Francesco Belardinelli, Wojciech Jamroga, Damian Kurpiewski, Vadim Malvone, and Aniello Murano. **Strategy Logic with Simple Goals: Tractable Reasoning about Strategies**. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI*, pages 88–94, 2019.

- (Kurpiewski et al. (2019c)) Damian Kurpiewski, Michal Knapik, and Wojciech Jamroga. **On Domination and Control in Strategic Ability (Extended Abstract)**. In *Proceedings of the 31st Benelux Conference on Artificial Intelligence (BNAIC)*, 2019.

- (Jamroga et al. (2018a)) W. Jamroga, M. Knapik, and D. Kurpiewski. **Model Checking the SELENE E-Voting Protocol in Multi-Agent Logics**. In *Proceedings of the 3rd International Joint Conference on Electronic Voting (E-VOTE-ID)*, **11143** of *Lecture Notes in Computer Science*, pages 100–116. Springer, 2018.

- (Jamroga et al. (2017a)) W. Jamroga, M. Knapik, and D. Kurpiewski. **Fixpoint Approximation of Strategic Abilities under Imperfect Information**. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1241–1249. IFAAMAS, 2017.

My scholarly contributions to date encompass a total of 21 conference papers, with 10 of these papers being accepted and published in A*-rated conferences, and an additional 3 published in B-rated conferences. Additionally, my research has been further disseminated through 4 journal papers, showcasing the breadth and depth of my work in the field.

2

Towards Practical Model Checking of Sociotechnical Systems

This chapter provides a comprehensive introduction to the theoretical foundations of our study, establishing a rigorous framework for the subsequent discussion. We begin by presenting a formal definition of the model-checking problem, which serves as the basis for our investigation. Within this framework, we clearly delineate the specific task at hand, providing a concise overview of the research objectives.

Following the formal definition, we provide an in-depth examination of potential solutions to the model-checking problem, highlighting relevant tools and methodologies that underpin our approach. This exposition is complemented by a practical dimension, where we present illustrative examples of tools that facilitate the definition and analysis of models. These examples are designed to engage the reader, providing a tangible understanding of the tools and techniques discussed.

By integrating theoretical foundations with practical applications, this chapter aims to provide a thorough yet accessible introduction to the fundamental concepts that underpin our research. The content of this chapter is informed by the following papers: Jamroga et al. (2019a, 2020b, 2022a).

2.1 Model-Checking Strategic Ability Under Imperfect Information

In this section we provide an overview of the relevant variants of **ATL**, and the corresponding complexity results for model checking.

The central operator of **ATL**, denoted as $\langle\langle A\rangle\rangle\gamma$, captures the notion that coalition A possesses a strategy to guarantee the satisfaction of the temporal property γ. Over the past 15 years, numerous semantic variants of **ATL** have emerged, exhibiting significant diversity in their underlying assumptions regarding the agents' memory and observational capabilities, as well as their interpretation of the concept of *ability*. This dichotomy is reflective of the rich and ongoing discourse in modern philosophy and AI, as exemplified by prominent works such as Belnap and Perloff (1988); McCarthy and Hayes (1969); Moore (1977, 1985); Ryle (1949), among others. In this context, we concentrate on Schobbens' $\mathbf{ATL}_{\text{ir}}$ as the definitive logic of strategic ability under imperfect information, wherein the existence of *memoryless* conditional plans is of primary concern. Furthermore, a plan is deemed successful if it achieves its objectives from all states that the coalition considers possible in the current state, thereby embodying the concept of *subjective ability*.

2.1.1 Models, Strategies, Outcomes

Models. We define the semantics of **ATL** specifications over a variant of transition systems, where transitions are labeled by combinations of actions, one per agent. Additionally, epistemic relations are employed to indicate states that are indistinguishable to a given agent. Formally, an *imperfect information concurrent game structure* or *iCGS* Alur et al. (2002); Schobbens (2004); van der Hoek and Wooldridge (2002) is represented as $M = \langle \mathbb{A}\text{gt}, St, Props, V, Act, d, o, \{\sim_a | \ a \in \mathbb{A}\text{gt}\}\rangle$, comprising:

- a nonempty finite set of agents $\mathbb{A}\text{gt} = \{1, \ldots, k\}$,
- a nonempty set of states St,
- a set of atomic propositions $Props$ and their valuation $V\colon Props \to 2^{St}$,
- a nonempty finite set of (atomic) actions Act,

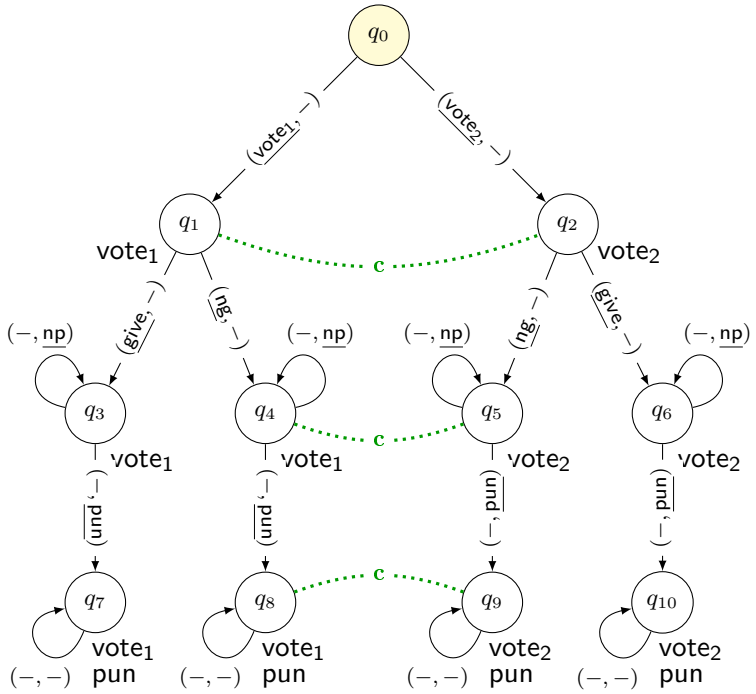

Figure 2.1: A simple model of voting and coercion

- a repertoire function d: $\mathbb{A}\mathrm{gt} \times St \to 2^{Act} \setminus \{\emptyset\}$, defining nonempty sets of actions available to agents at each state; we denote $d_a(q)$ instead of $d(a, q)$, and define $d_A(q) = \prod_{a \in A} d_a(q)$ for each $A \subseteq \mathbb{A}\mathrm{gt}, q \in St$,
- a (deterministic) transition function o that assigns the outcome state $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to each state q and tuple of actions $\langle \alpha_1, \ldots, \alpha_k \rangle$ such that $\alpha_i \in d(i, q)$ for $i = 1, \ldots, k$ that can be executed by $\mathbb{A}\mathrm{gt}$ in q.

Furthermore, every $\sim_a \subseteq St \times St$ is an epistemic equivalence relation, implying that states q and q' are indistinguishable to agent a whenever $q \sim_a q'$. We assume the iCGS to be *uniform*, meaning that $q \sim_a q'$ implies $d_a(q) = d_a(q')$, i.e., the same choices are available in indistinguishable states. Notably, perfect information can be modeled by assuming each $\sim_a$ to be the identity relation.

Example 2.1.1 *Consider a simple voting scenario involving two agents: the voter v and the coercer c. The voter casts a vote for a selected candidate $i \in \{1, \ldots, n\}$ by performing the action vote$_i$. Upon exiting the polling station, the voter has the option to hand in a proof of their vote to the coercer (action give) or refuse to provide the*

proof (action ng). The proof can take various forms, such as a certified receipt from the election authorities or a picture of the ballot taken with a smartphone, which the coercer will consider believable. Following this, the coercer can either punish the voter (action pun) or refrain from punishment (action np).

The iCGS M_{vote} modeling this scenario for $n = 2$ is depicted in Figure 2.1. In this model, we employ two propositional variables:

- vote_i *labels states where the voter has already cast their vote for candidate i,*
- pun *indicates states where the voter has been punished.*

Notably, the coercer remains unaware of the voter's choice unless the voter decides to provide the proof. This lack of knowledge defines the indistinguishability relation for the coercer, which is represented by dotted lines in the model.

Strategies. A *strategy* of agent $a \in \mathbb{A}\mathrm{gt}$ is a conditional plan that specifies the actions to be taken by a in every possible situation. Formally, a *perfect information memoryless strategy* for a can be represented by a function $s_a \colon St \to Act$ that satisfies $s_a(q) \in d_a(q)$ for each $q \in St$. In contrast, an *imperfect information memoryless strategy* also satisfies the condition $s_a(q) = s_a(q')$ whenever $q \sim_a q'$, indicating that the strategy is invariant under indistinguishable states. Following the terminology of Schobbens (2004), we refer to the former as Ir-*strategies* and to the latter as ir-*strategies*.

A *collective x-strategy* s_A for coalition $A \subseteq \mathbb{A}\mathrm{gt}$ and strategy type $x \in \{\mathrm{Ir}, \mathrm{ir}\}$ is a tuple of individual x-strategies, one per agent in A. The set of all such strategies is denoted by Σ^x_A. By $s_A|_a$, we denote the strategy of agent $a \in A$ selected from s_A.

To facilitate the discussion of partial strategies, we introduce the notion of function extension. Given two partial functions $f, f' \colon X \rightharpoonup Y$, we say that f' *extends* f (denoted $f \subseteq f'$) if, whenever $f(x)$ is defined, we have $f(x) = f'(x)$. A partial function $s'_a \colon St \rightharpoonup Act$ is called *a partial x-strategy for a* if s'_a is extended by some strategy $s_a \in \Sigma^x_a$. A collective partial x-strategy s_A is a tuple of partial x-strategies, one per agent in A.

Outcome paths. A *path* $\lambda = q_0 q_1 q_2 \ldots$ is an infinite sequence of states such that there is a transition between each q_i, q_{i+1}. We use $\lambda[i]$ to denote the ith position on path λ (starting from $i = 0$) and $\lambda[i, j]$ to denote the part of λ between positions i and j. The function $out(q, s_A)$ returns the set of all paths that can result from the execution of a (complete) strategy s_A, beginning at state q. For agents not in A, path transitions can involve any action allowed by the protocol function.

Formally, the function $out(q, s_A)$ is defined as:

$out(q, s_A) = \{\lambda = q_0, q_1, q_2 \ldots \mid q_0 = q$ and for each $i = 0, 1, \ldots$ there exists $\langle \alpha^i_{a_1}, \ldots, \alpha^i_{a_k} \rangle$ such that $\alpha^i_a \in d_a(q_i)$ for every $a \in \mathbb{A}\mathrm{gt}$, and $\alpha^i_a = s_A|_a(q_i)$ for every $a \in A$, and $q_{i+1} = o(q_i, \alpha^i_{a_1}, \ldots, \alpha^i_{a_k})\}$.

We will sometimes write $out^{\mathrm{Ir}}(q, s_A)$ instead of $out(q, s_A)$. Furthermore, we define the function $out^{\mathrm{ir}}(q, s_A) = \bigcup_{a \in A} \bigcup_{q \sim_a q'} out(q', s_A)$, which collects all the outcome paths that start from states that are indistinguishable from q to at least one agent in A. This allows us to account for the imperfect information that agents may have about the initial state of the interaction.

2.1.2 Alternating-Time Temporal Logic

Syntax. We employ a variant of **ATL** that explicitly distinguishes between perfect and imperfect information abilities. The syntax of this variant is defined by the following grammar Alur et al. (1997):

$$\begin{aligned} \varphi &::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A \rangle\!\rangle_x \gamma \\ \gamma &::= \mathbf{X}\varphi \mid \mathbf{G}\varphi \mid \varphi\, \mathbf{U}\, \varphi, \end{aligned}$$

where $x \in \{\mathrm{Ir}, \mathrm{ir}\}$, $\mathsf{p} \in \mathit{Props}$ and $A \subseteq \mathbb{A}\mathrm{gt}$. Formulae γ are sometimes referred to as *path subformulae* of **ATL**. The formula $\langle\!\langle A \rangle\!\rangle_{\mathrm{ir}} \gamma$ is read as *"A can identify and execute a strategy that enforces γ,"* while $\mathbf{X}$ is interpreted as *"in the next state,"* $\mathbf{G}$ as *"now and always in the future,"* and $\mathbf{U}$ as *"until."*

In contrast, the perfect information modality $\langle\!\langle A \rangle\!\rangle_{\mathrm{Ir}} \gamma$ can be understood as *"A might be able to bring about γ if allowed to make lucky guesses whenever uncertain."* Our primary focus is on the type of ability expressed by $\langle\!\langle A \rangle\!\rangle_{\mathrm{ir}}$, which represents the ability of a coalition to achieve a goal under imperfect information. The other strategic modality (i.e., $\langle\!\langle A \rangle\!\rangle_{\mathrm{Ir}}$) will prove useful in approximating $\langle\!\langle A \rangle\!\rangle_{\mathrm{ir}}$.

Semantics. The semantics of **ATL** can be defined as follows:

- $M, q \models \mathsf{p}$ iff $q \in V(\mathsf{p})$,
- $M, q \models \neg\varphi$ iff $M, q \not\models \varphi$,

- $M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$,
- $M, q \models \langle\!\langle A \rangle\!\rangle_x \mathbf{X}\varphi$ iff there exists $s_A \in \Sigma_A^x$ such that for all $\lambda \in out^x(q, s_A)$ we have $M, \lambda[1] \models \varphi$,
- $M, q \models \langle\!\langle A \rangle\!\rangle_x \mathbf{G}\varphi$ iff there exists $s_A \in \Sigma_A^x$ such that for all $\lambda \in out^x(q, s_A)$ and $i \in \mathbb{N}$ we have $M, \lambda[i] \models \varphi$,
- $M, q \models \langle\!\langle A \rangle\!\rangle_x \psi \,\mathbf{U}\, \varphi$ iff there exists $s_A \in \Sigma_A^x$ such that for all $\lambda \in out^x(q, s_A)$ there is $i \in \mathbb{N}$ for which $M, \lambda[i] \models \varphi$ and $M, \lambda[j] \models \psi$ for all $0 \leq j < i$.

The standard boolean operators, including the logical constants $\top$ and $\bot$, disjunction $\vee$, and implication $\rightarrow$, are defined in the usual manner. For convenience, we will often write $\langle A \rangle \varphi$ instead of $\langle\!\langle A \rangle\!\rangle_{\text{ir}} \mathbf{X}\varphi$ to express one-step abilities under imperfect information. Furthermore, we define the operator *"now or sometime in the future"* as $\mathbf{F}\varphi \equiv \top \,\mathbf{U}\, \varphi$.

It is straightforward to observe that $M, q \models \langle\!\langle A \rangle\!\rangle_x \mathbf{F}\varphi$ holds if and only if there exists a collective strategy $s_A \in \Sigma_A^x$ such that, on each path $\lambda \in out^x(q, s_A)$, there is a state satisfying φ. In this case, we can also say that φ is *x-reachable from q for A*, indicating that the coalition A has the ability to reach a state satisfying φ from the current state q under the specified information type x.

Example 2.1.2 *Consider the model M_{vote} from Example 2.1.1. The formula $\varphi_0 \equiv \langle\!\langle c \rangle\!\rangle_{\text{ir}} \mathbf{F}\big(\neg \mathsf{vote_i} \rightarrow \mathsf{pun}\big)$ expresses that the coercer can ensure that the voter will eventually either have voted for candidate i (presumably chosen by the coercer for the voter to vote for) or be punished. Notably, this formula holds in M_{vote}, q_0 for any $i = 1, 2$.*

A strategy for the coercer c that witnesses this property is defined as follows: $s_c(q_3) = np, s_c(q_4) = s_c(q_5) = s_c(q_6) = pun$ for $i = 1$, and symmetrically for $i = 2$. This strategy enables the coercer to enforce the desired outcome, regardless of the voter's actions.

As a consequence, the formula $\varphi_1 \equiv \langle\!\langle v \rangle\!\rangle_{\text{ir}} \mathbf{G}\big(\neg \mathsf{pun} \wedge \neg \mathsf{vote_i}\big)$, which states that the voter can avoid voting for candidate i and being punished, is false in M_{vote}, q_0 for all $i = 1, 2$. This indicates that the voter does not have a strategy to guarantee the avoidance of punishment and voting for the unwanted candidate.

Complexity. We distinguish between two syntactic fragments of the logic: $\mathbf{ATL}_{\text{ir}}$, which contains only $\langle\!\langle A \rangle\!\rangle_{\text{ir}}$ modalities, and $\mathbf{ATL}_{\text{Ir}}$, which contains only $\langle\!\langle A \rangle\!\rangle_{\text{Ir}}$ modalities.

The complexity of model checking for these fragments is established by the following proposition:

Proposition 2.1.3 (Alur et al. (2002); Jamroga and Dix (2006); Schobbens (2004)) *Model checking* $\mathbf{ATL}_{\mathrm{Ir}}$ *is* $\mathbf{P}$*-complete and can be done in time* $O(|M| \cdot |\varphi|)$ *where* $|M|$ *is the number of transitions in the model and* $|\varphi|$ *is the length of the formula. Model checking* $\mathbf{ATL}_{\mathrm{ir}}$ *is* $\mathbf{\Delta_2^P}$*-complete with respect to* $|M|$ *and* $|\varphi|$*.*

This proposition highlights the difference in complexity between model checking for $\mathbf{ATL}_{\mathrm{Ir}}$ and $\mathbf{ATL}_{\mathrm{ir}}$, with the former being tractable in polynomial time and the latter being more computationally demanding due to its $\mathbf{\Delta_2^P}$-completeness. $\mathbf{\Delta_2^P}$ is a class within the polynomial hierarchy that consists of decision problems solvable in polynomial time by a deterministic Turing machine equipped with an oracle for an NP-complete problem. Formally, $\mathbf{\Delta_2^P} = P^{NP}$, meaning it captures the complexity of problems that can be solved with a polynomial-time algorithm that can make a bounded number of calls to a subroutine that solves any problem in NP.

2.1.3 Reasoning about Knowledge

Given the indistinguishability relations in the models, we can interpret knowledge modalities K_a in a standard manner:

$M, q \models K_a\varphi$ iff $M, q' \models \varphi$ for all q such that $q \sim_a q'$.

Intuitively, the meaning of $q \models K_a\varphi$ is that the observational capabilities of agent a allow the agent to conclude that φ holds in each state that is indistinguishable from q according to a.

We also define the semantics of *"everybody knows" (E_A)* and *common knowledge (C_A)* in a similar manner. To aggregate individual uncertainty in A, we assume the relation $\sim_A^E = \bigcup_{a\in A} \sim_a$. For common knowledge, we take $\sim_A^C$ to be the transitive closure of $\sim_A^E$. By convention, we take $\sim_\emptyset^E$ and $\sim_\emptyset^C$ to be the identity relations.

Furthermore, we use the notation $[q]_{\mathcal{R}} = \{q' \mid q\mathcal{R}q'\}$ to denote the image of q with respect to relation $\mathcal{R}$. This notation provides a concise way to express the set of states that are related to q via the relation $\mathcal{R}$.

Example 2.1.4 *The following formulae hold in* M_{vote}, q_0 *for any* $i = 1, 2$ *due to the strategy* s_c *presented in Example 2.1.2:*

- $\varphi_2 \equiv \langle\!\langle c \rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\big((\neg K_c \mathsf{vote_i}) \rightarrow \mathsf{pun}\big)$*: This formula states that the coercer has a strategy to ensure that, eventually, the voter is punished unless the coercer has learned that the voter voted as instructed.*

- $\varphi_3 \equiv \langle\!\langle c \rangle\!\rangle_{\text{ir}} \mathbf{G}\big((K_c \mathsf{vote_i}) \to \neg \mathsf{pun}\big)$*: Furthermore, the coercer can guarantee that if he learns that the voter obeyed, then the voter will not be punished.*

It is worth noting that the property expressed by the first formula imposes a rather strict requirement on the coercer's expectations. Specifically, in a system where the formula holds, the coercer enforces a strategy that punishes the voter at every reached state where he is not certain *that the voter voted for the i-th candidate. Since* $q \models \neg K_c \mathsf{vote_i} \to \mathsf{pun}$ *implies* $q \models \neg \mathsf{vote_i} \to \mathsf{pun}$*, we observe that* φ_2 *implies* φ_0 *of Example 2.1.2. This indicates that the coercer's strategy to punish the voter unless he is certain of the voter's obedience is a more stringent requirement than simply punishing the voter unless he votes as instructed.*

Remark 2.1.5 *It is worth noting that* $K_a\varphi$ *can be defined in* $\mathbf{ATL}_{\text{ir}}$ *as* $\langle\!\langle a \rangle\!\rangle_{\text{ir}} \bot \mathbf{U}\, \varphi$*. Furthermore, the notion of "everybody knows" can be entirely defined in* $\mathbf{ATL}_{\text{ir}}$ *by* $E_A\varphi \equiv \langle\!\langle A \rangle\!\rangle_{\text{ir}} \bot \mathbf{U}\, \varphi$*. This highlights the connection between knowledge and strategic abilities in the context of* $\mathbf{ATL}_{\text{ir}}$*.*

Remark 2.1.6 *The semantics of* $\langle\!\langle A \rangle\!\rangle_{\text{ir}} \gamma$*, as presented in Section 2.1.2, encodes the notion of "subjective" ability Jamroga and van der Hoek (2004); Schobbens (2004). This means that the agents must possess a successful strategy from all the states that they consider possible when the system is in state q. Consequently, they know that the strategy indeed obtains* γ*. In contrast, the alternative notion of "objective" ability Bulling and Jamroga (2014) requires the existence of a winning strategy from state q alone.*

We focus on the subjective interpretation, as it is more standard in game theory and **ATL***. This is primarily because it formalizes the notion of "knowing how to play" in a more relevant manner for game solving. To illustrate this, consider a card game such as poker or bridge, where the challenge is to find a strategy that wins for all possible hands of the opponents.*

An alternative perspective involves considering objective ability*, which focuses solely on the outcome paths originating from the current global state of the system. Interestingly, the frameworks of subjective and objective ability share a similar structure, allowing for conversions between the two by modifying the underlying model. Specifically, transitioning from objective ability to subjective ability can be accomplished by introducing a new initial state with a single outgoing transition that leads to the "real" initial state. Conversely, converting from subjective ability to objective ability entails adding a new initial state with multiple outgoing transitions, controlled by the Environment, each of which directs to a distinct state within the initial epistemic class. This highlights the*

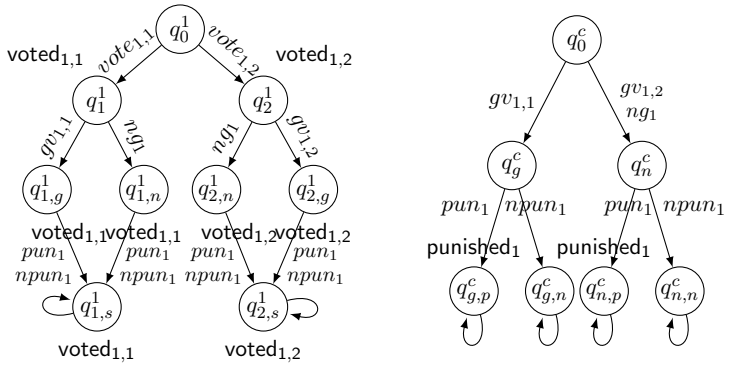

Figure 2.2: ASV_1^2: agents *Voter*$_1$ (left) and *Coercer* (right)

versatility of these approaches and facilitates a deeper understanding of the relationship between subjective and objective ability.

It is worth noting that if $[q]_{\sim_A^E} = \{q\}$ *and* γ *contains no nested strategic modalities, then the subjective and objective semantics of* $\langle\langle A\rangle\rangle_{\mathrm{ir}}\gamma$ *at* q *coincide. Therefore, the subjective and objective interpretation of the formula in Example 2.1.2 is actually the same.*

Moreover, if $\varphi, \varphi_1, \varphi_2$ *contain no nested strategic modalities, then model checking* $\langle\langle A\rangle\rangle_{\mathrm{ir}}\varphi_1 \mathbf{U} \varphi_2$ *and* $\langle\langle A\rangle\rangle_{\mathrm{ir}}\mathbf{G}\varphi$ *in* M, q *according to the objective semantics can be easily reduced to the subjective case by adding a spurious initial state* q'*, with transitions to all states in* $[q]_{\sim_A^E}$*, controlled by a "dummy" agent outside* A*. For the details of this construction, we refer the interested reader to Pilecki et al. (2017).*

2.1.4 Models of Asynchronous Interaction

Definition 2.1.7 (Asynchronous MAS) *An* asynchronous multi-agent system (AMAS) *S consists of n agents* $\mathrm{A} = \{1, \ldots, n\}$*, each associated with a tuple* $A_i = (L_i, \iota_i, Evt_i, R_i, T_i, PV_i, V_i)$ *including a set of* local states $L_i = \{l_i^1, l_i^2, \ldots, l_i^{n_i}\}$*, a designated* initial state $\iota_i \in L_i$*, a nonempty finite set of* events $Evt_i = \{\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{m_i}\}$*, and a* repertoire of choices $R_i : L_i \to 2^{2^{Evt_i}}$*. For each* $l_i \in L_i$*,* $R_i(l_i) = \{E_1, \ldots, E_m\}$ *is a nonempty list of nonempty choices available to* i *at* l_i*. If the agent chooses* $E_j = \{\alpha_1, \alpha_2, \ldots\}$*, then only an event in* E_j *can be executed at* l_i *within the agent's module. Moreover,* $T_i \subseteq L_i \times Evt_i \times L_i$ *is a* local transition relation*, where* $(l_i, \alpha, l_i') \in T_i$ *represents that event* $\alpha \in \bigcup R_i(l_i)$ *changes the local state from* l_i *to* l_i'*. Agents are endowed with mutually disjoint, finite and possibly empty sets of* local propositions PV_i*, and their* valuations $V_i : L_i \to 2^{PV_i}$*.*

Note that each agent "owns" the events affecting its state, but some of the events

may be shared with other agents. Those can only be executed synchronously by all the involved parties.

This way, the agent can influence how the states of the other agents evolve.

Moreover, the agent's strategic choices are restricted by its repertoire function. Assigning sets rather than single events in R_i is a deliberate decision, allowing to avoid certain semantic issues that fall outside the scope of this thesis. We refer the reader to Jamroga et al. (2021) for the details.

The following example demonstrates a simple AMAS, while also introducing some key concepts. In particular, there is a *coercer* agent, whose goal is to ensure that the voter(s) select a particular candidate. To that end, the coercer may threaten them with punishment, e.g. if they refuse to cooperate by not sharing the ballot, or openly defy by voting for another candidate.

Example 2.1.8 (Asynchronous Simple Voting) *Consider a simple voting system* ASV_n^k *with* $n+1$ *agents (*n *voters and* 1 *coercer). Each* $Voter_i$ *agent can cast her vote for a candidate* $\{1,\dots,k\}$*, and decide whether to share her vote receipt with the Coercer agent. The coercer can choose to punish the voter or refrain from it. A graphical representation of the agents for* $n=1, k=2$ *is shown in Fig. 2.2. We assume that the coercer only registers if the voter hands in a receipt for candidate* 1 *or not. The repertoire of the coercer is defined as* $R_c(q_0^c) = \{\{gv_{1,1}, gv_{1,2}, ng_1\}\}$ *and* $R_c(q_g^c) = R_c(q_n^c) = \{\{pun_1\},\{npun_1\}\}$*, i.e., the coercer first* receives *the voter's decision regarding the receipt, and then* controls *whether the voter is punished or not. Analogously, the voter's repertoire is given by:* $R_1(q_0^1) = \{\{vote_{1,1}\},\{vote_{1,2}\}\}$*,* $R_1(q_j^1) = \{\{gv_{1,j}\},\{ng_1\}\}$ *for* $j=1,2$*, and* $R_1(q_{1,g}^1) = R_1(q_{1,n}^1) = R_1(q_{2,g}^1) = R_1(q_{2,n}^1) = \{\{pun_1, npun_1\}\}$.

Notice that the coercer cannot determine which of the events $gv_{1,1}, gv_{1,2}, ng_1$ will occur; this is entirely under the voter's control. This way we model the situation where it is the decision of the voter to show her vote or not. Similarly, the voter cannot avoid punishment by choosing the strategy allowing only $npun_1$, because the choice $\{npun_1\}$ is *not* in the voter's repertoire. She can only execute $\{pun_1, npun_1\}$, and await the decision of the coercer.

The execution semantics is based on interleaving with synchronization on shared events. Note that for a shared event to be executed, it must be done jointly by all agents who have it in their repertoires.

Definition 2.1.9 (Interleaved interpreted system) *Let S be an AMAS with n agents. The* interleaved interpreted system *$IIS(S)$ extends S with: (i) the initial states $\iota = (\iota_1, \ldots, \iota_n)$; (ii) the set of global states $St \subseteq L_1 \times \ldots \times L_n$ that collects all the configurations of local states, reachable from ι by T (see below); (iii) the (partial)* global transition function *$T : St \times Evt \rightharpoonup St$, defined by $T(q_1, \alpha) = q_2$ iff $T_i(q_1^i, \alpha) = q_2^i$ for all $i \in Agent(\alpha)$ and $q_1^i = q_2^i$ for all $i \in \mathrm{A} \setminus Agent(\alpha)$;*[1] *(iv) the* global valuation *of propositions $V : St \to 2^{PV}$, defined as $V(l_1, \ldots, l_n) = \bigcup_{i \in \mathrm{A}} V_i(l_i)$.*
We will sometimes write $q_1 \xrightarrow{\alpha} q_2$ instead of $T(q_1, \alpha) = q_2$. Also, we define relation $\sim_A = \{(q, q') \in St \times St \mid \exists i \in A \,.\, q^i = q'^i\}$ to connect states that are indistinguishable for at least one agent $i \in A$.

Definition 2.1.10 (Enabled events) *Let $A = \{a_1, \ldots, a_k\} \subseteq \mathrm{A} = \{1, \ldots, n\}$ and $\overrightarrow{E}_A = (E_{a_1}, \ldots, E_{a_k})$ for some $k \leq n$, such that $E_i \in R_i(q^i)$ for every $i \in A$. Event $\beta \in Evt$ is* enabled by the vector of choices $\overrightarrow{E}_A$ at $q \in St$ *iff, for every $i \in Agent(\beta) \cap A$, we have $\beta \in E_i$, and for every $i \in Agent(\beta) \setminus A$, it holds that $\beta \in \bigcup R_i(q^i)$. That is, the "owners" of β in A have selected choices that admit β, while all the other "owners" of β* might *select choices that do the same. We denote the set of such events by $enabled(q, \overrightarrow{E}_A)$.*

Some combinations of choices enable no events. To account for this, the models of AMAS are augmented with "silent" ϵ-loops, added when no "real" event can occur.

Definition 2.1.11 (Undeadlocked IIS) *Let S be an AMAS, and assume that no agent in S has ϵ in its alphabet of events. The* model of S, *denoted $IIS^{\epsilon}(S, I)$, extends the model $IIS(S, I)$ as follows:*

- *$Evt_{IIS^{\epsilon}(S)} = Evt_{IIS(S)} \cup \{\epsilon\}$, where $Agent(\epsilon) = \emptyset$;*
- *For each $q \in St$, we add the transition $q \xrightarrow{\epsilon} q$ iff there is a selection of all agents' choices $\overrightarrow{E}_{\mathbb{A}\mathrm{gt}} = (E_1, \ldots, E_n)$, such that $E_i \in R_i(q^i)$ and $enabled_{IIS(S,I)}(q, \overrightarrow{E}_{\mathbb{A}\mathrm{gt}}) = \emptyset$. Then, for every $A \subseteq \mathbb{A}\mathrm{gt}$, we also fix $enabled_{IIS^{\epsilon}(S)}(q, \overrightarrow{E}_A) = enabled_{IIS^{\epsilon}(S)}(q, \overrightarrow{E}_A) \cup \{\epsilon\}$.*

  In other words, an ϵ-loop is enabled whenever E_A allows the grand coalition to collectively block the execution of any "real" event.

Example 2.1.12 *The model of ASV_1^2 is shown in Figure 2.3. Note that it contains no ϵ-transitions, since no choices of the voter and the coercer can cause a deadlock.*

[1] q^i denotes agent i's state in $q = (l_1, \ldots, l_n)$, i.e., $q^i = l_i$.

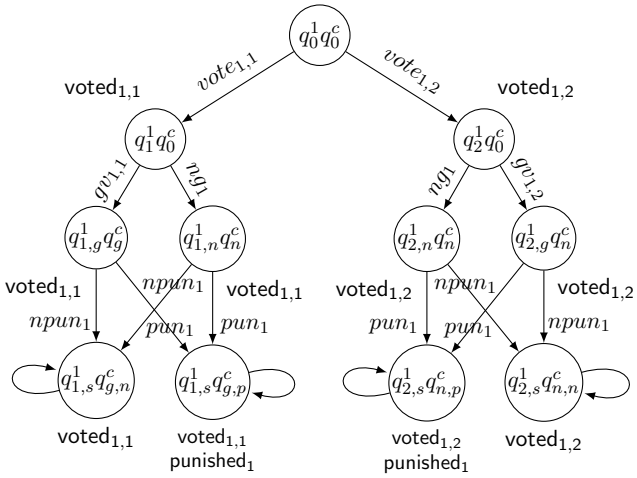

Figure 2.3: The model $IIS^{\epsilon}(ASV_1^2)$

2.1.5 Reasoning About Strategies

Strategic ability of agents. Following Jamroga et al. (2021), a *positional imperfect information strategy (*ir*-strategy) for agent i* is defined by a function $\sigma_i \colon L_i \to 2^{Evt_i}$, such that $\sigma_i(l) \in R_i(l)$ for each $l \in L_i$. Note that σ_i is uniform by construction, as it is based on local, and not global states. Moreover, every uniform strategy in the (global) model can be represented by a function of type $L_i \to 2^{Evt_i}$. Thus, we can also denote the set of such strategies by Σ_i^{ir}. Joint strategies Σ_A^{ir} for $A = \{a_1, \ldots, a_k\} \subseteq \mathrm{A}$ are defined as usual, i.e., as tuples of strategies σ_i, one for each agent $i \in A$. By $\sigma_A(q) = (\sigma_{a_1}(q), \ldots, \sigma_{a_k}(q))$, we denote the joint choice of coalition A at global state q. An infinite sequence of global states and events $\pi = q_0\alpha_0 q_1 \alpha_1 q_2 \ldots$ is called a *path* if $q_j \xrightarrow{\alpha_j} q_{j+1}$ for every $j \geq 0$. The set of all paths in model M starting at state q is denoted by $\Pi_M(q)$.

Definition 2.1.13 (Standard outcome) *Let $A \subseteq \mathrm{A}$. The* standard outcome *of strategy $\sigma_A \in \Sigma_A^{\mathrm{ir}}$ in state q of model M is the set $out_M^{\mathrm{Std}}(q, \sigma_A) \subseteq \Pi_M(q)$ such that $\pi = q_0\alpha_0 q_1 \alpha_1 \cdots \in out_M(q, \sigma_A)$ iff $q \sim_j q_0$, and for each $m \geq 0$ we have that $\alpha_m \in enabled_M(q_m, \sigma_A(q_m))$.*

Definition 2.1.14 (Reactive outcome)

The reactive outcome *is the set $out_M^{\mathrm{React}}(q, \sigma_A) \subseteq out_M^{\mathrm{Std}}(q, \sigma_A)$ such that $\pi = q_0\alpha_0 q_1 \alpha_1 \cdots \in out_M^{\mathrm{React}}(q, \sigma_A)$ iff $\alpha_m = \epsilon$ implies $enabled_M(q_m, \sigma_A(q_m)) = \{\epsilon\}$.*

Intuitively, the standard outcome collects all the paths where agents in A follow σ_A, while the others freely choose from their repertoires. The reactive outcome includes only those outcome paths where the opponents cannot miscoordinate on shared events. Let $x \in \{\text{Std}, \text{React}\}$. We extend the above definitions to subsets of global states $G \subseteq St$ by $out^x_M(G, \sigma_A) = \bigcup_{q \in G} out^x_M(q, \sigma_A)$. Now, the ir-semantics of **ATL** in asynchronous MAS Alur et al. (2002); Jamroga et al. (2021); Schobbens (2004) is defined by:

$M, q \models^x \mathsf{p}$ iff $q \in V(\mathsf{p})$, for $\mathsf{p} \in \mathit{Props}$;

$M, q \models^x \neg\varphi$ iff $M, q \not\models^x \varphi$;

$M, q \models^x \varphi_1 \wedge \varphi_2$ iff $M, q \models^x \varphi_1$ and $M, q \models^x \varphi_2$;

$M, q \models^x \langle\!\langle A \rangle\!\rangle \mathbf{X}\varphi$ iff there is a strategy $s_A \in \Sigma^{ir}_A$ such that, for each path $\lambda \in out^x_M([q]_{\sim_A}, s_A)$, we have $M, \lambda[1] \models^x \varphi$.

$M, q \models^x \langle\!\langle A \rangle\!\rangle \mathbf{G}\varphi$ iff there is a strategy $s_A \in \Sigma^{ir}_A$ such that, for each $\lambda \in out^x_M([q]_{\sim_A}, s_A)$ and $i \geq 0$, we get $M, \lambda[i] \models^x \varphi$.

$M, q \models^x \langle\!\langle A \rangle\!\rangle \varphi_1 \,\mathbf{U}\, \varphi_2$ iff there is a strategy $s_A \in \Sigma^{ir}_A$ such that, for each path $\lambda \in out^x_M([q]_{\sim_A}, s_A)$, we have $M, \lambda[i] \models^x \varphi_2$ for some $i \geq 0$ and $M, \lambda[j] \models^x \varphi_1$ for all $0 \leq j < i$.

Example 2.1.15 *Let $M = IIS^{\epsilon}(ASV^k_n)$, i.e., the model of the AMAS in Example 2.1.8. Note that the* Std *and* React *semantics coincide on M, as it includes no ϵ-transitions. Clearly, $M, (q^1_0, \ldots, q^n_0, q^c_0) \models \bigwedge_{æ \in Candidates} \langle\!\langle \mathit{Voter} \rangle\!\rangle \mathbf{F}\mathsf{voted}_{\mathsf{j}}$.*

In this paper, we focus on formulas with no next step operators **X** and no nested strategic modalities. The corresponding "simple" subset of **ATL** (resp. **ATLK**) is denoted by **sATL** (resp. **sATLK**). The restriction is less prohibitive than it seems at a glance. First, the **X** operator is of little value for asynchronous systems. Secondly, nested strategic modalities would only allow us to express an agent's ability to endow another agent with ability (or deprive the other agent of ability). Such properties are sometimes relevant, e.g., one may want to require that $\langle\!\langle \mathit{Voter}_1 \rangle\!\rangle \mathbf{G} \neg \langle\!\langle \mathit{Coercer} \rangle\!\rangle \mathbf{F}\mathsf{punished}_1$ (the voter can keep the coercer unable to punish the voter). Still, simpler properties like $\langle\!\langle \mathit{Voter}_1 \rangle\!\rangle \mathbf{G} \neg \mathsf{punished}_1$ and $\langle\!\langle \mathit{Voter}_1 \rangle\!\rangle \mathbf{G} \bigwedge_{j=1,\ldots,k} \neg K_{\mathit{Coercer}} \mathsf{voted}_{1,\mathsf{j}}$ are usually of more immediate interest.

2.1.6 Alternating Epistemic Mu-Calculus

The modalities in $\mathbf{ATL}_{\mathrm{Ir}}$ are known to have straightforward fixpoint characterizations Alur et al. (2002), enabling the embedding of $\mathbf{ATL}_{\mathrm{Ir}}$ in a variant of μ-calculus. This variant utilizes $\langle\langle A\rangle\rangle_{\mathrm{Ir}}\mathbf{X}$ as the basic modality and does not involve alternation of fixpoint operators.

In contrast, the analogous variant of μ-calculus for imperfect information exhibits incomparable expressive power to $\mathbf{ATL}_{\mathrm{ir}}$ Bulling and Jamroga (2011). This disparity suggests that, under imperfect information, **ATL** and fixpoint specifications offer distinct perspectives on strategic ability.

This section provides a concise overview of the syntax and semantics of this μ-calculus variant, as well as a recap of the pertinent results.

Formally, the *alternating epistemic μ-calculus ($\mathbf{AE}\mu\mathbf{C}$)* is defined by augmenting the next-time fragment of $\mathbf{ATL}_{\mathrm{ir}}$, possibly with epistemic modalities, with the least fixpoint operator μ. The greatest fixpoint operator ν is introduced as the dual to μ.

Let $\mathcal{V}ars$ denote a set of second-order variables ranging over 2^{St}. The language of $\mathbf{AE}\mu\mathbf{C}$ is defined by the following grammar:

$$\varphi ::= \mathsf{p} \mid Z \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle A\rangle\varphi \mid \mu Z(\varphi) \mid K_a\varphi,$$

where $\mathsf{p} \in \mathit{Props}$, $Z \in \mathcal{V}ars$, $a \in \mathbb{A}\mathrm{gt}$, $A \subseteq \mathbb{A}\mathrm{gt}$, and the formulae are Z–positive. This means that each free occurrence of Z is in the scope of an even number of negations.

We define the greatest fixpoint operator $\nu Z(\varphi(Z))$ as the dual to $\mu Z(\varphi(Z))$, i.e., $\nu Z(\varphi(Z)) \equiv \neg\mu Z(\neg\varphi(\neg Z))$, where $\varphi(\neg Z)$ denotes the result of substituting in φ all free occurrences of Z with $\neg Z$.

A formula of $\mathbf{AE}\mu\mathbf{C}$ is considered *simple* if, in its negation normal form, it contains no occurrences of ν (resp. μ) on any syntactic path from an occurrence of μZ (resp. νZ) to a bound occurrence of Z. In other words, simple formulae do not exhibit alternation of fixpoint operators.

We restrict our attention to the simple fragment of $\mathbf{AE}\mu\mathbf{C}$, denoted by $\mathbf{sAE}\mu\mathbf{C}$, as it simplifies the semantics of μ-calculus and typically reduces its model checking complexity. This approach is consistent with Bulling and Jamroga (2011), where the authors also focus on the simple fragment of $\mathbf{AE}\mu\mathbf{C}$.

We evaluate the formulae of $\mathbf{sAE}\mu\mathbf{C}$ with respect to valuations of $\mathcal{V}ars$, which are functions $\mathcal{V}\colon \mathcal{V}ars \to 2^{St}$. The set of all valuations of $\mathcal{V}ars$ is denoted by $\mathcal{V}als$.

Given a variable $X \in \mathcal{V}ars$, a set of states $Z \subseteq St$, and a valuation $\mathcal{V} \in \mathcal{V}als$, we define a new valuation $\mathcal{V}[X := Z]$ as follows:

- For any variable $Y \neq X$, $\mathcal{V}[X := Z](Y) = \mathcal{V}(Y)$, i.e., the valuation of Y remains unchanged.
- For the variable X, $\mathcal{V}[X := Z](X) = Z$, i.e., the valuation of X is updated to Z.

This notation allows us to concisely express the modification of a valuation by updating the value of a specific variable.

The denotational semantics of **sAEμC** assigns to each formula φ the set of states $[\![\varphi]\!]^M_{\mathcal{V}}$ where φ is true under the valuation $\mathcal{V} \in \mathcal{V}als$. The semantics is defined recursively as follows:

- For atomic propositions, $[\![\mathsf{p}]\!]^M_{\mathcal{V}} = V(\mathsf{p})$, where V is the valuation function of the model.
- For variables, $[\![Z]\!]^M_{\mathcal{V}} = \mathcal{V}(Z)$, where $\mathcal{V}$ is the valuation of the variables.
- For negation, $[\![\neg\varphi]\!]^M_{\mathcal{V}} = St \setminus [\![\varphi]\!]^M_{\mathcal{V}}$, where St is the set of all states.
- For conjunction, $[\![\varphi \wedge \psi]\!]^M_{\mathcal{V}} = [\![\varphi]\!]^M_{\mathcal{V}} \cap [\![\psi]\!]^M_{\mathcal{V}}$.
- For the next-time operator, $[\![\langle A\rangle\varphi]\!]^M_{\mathcal{V}} = \{q \in St \mid \exists s_A \in \Sigma_A\ \forall \lambda \in out^{\mathrm{ir}}_M(q, s_A)\ \lambda[1] \in [\![\varphi]\!]^M_{\mathcal{V}}\}$, where $out^{\mathrm{ir}}_M(q, s_A)$ is the set of all paths starting from q and conforming to strategy s_A.
- For the least fixpoint operator, $[\![\mu Z(\varphi)]\!]^M_{\mathcal{V}} = \bigcap\{Q \subseteq St \mid [\![\varphi]\!]^M_{\mathcal{V}[Z:=Q]} \subseteq Q\}$, where $\mathcal{V}[Z := Q]$ is the valuation obtained by updating the value of Z to Q.
- For the epistemic operator, $[\![K_a\varphi]\!]^M_{\mathcal{V}} = \{q \in St \mid \forall_{q'}(q \sim_a q' \text{ implies } q' \in [\![\varphi]\!]^M_{\mathcal{V}})\}$, where $\sim_a$ is the epistemic relation of agent a.

If φ is a sentence, i.e., it contains no free variables, then its validity does not depend on the valuation $\mathcal{V}$, and we write $M, q \models \varphi$ instead of $q \in [\![\varphi]\!]^M_{\mathcal{V}}$. This notation indicates that φ is true at state q in model M.

Example 2.1.16 *Consider the* **AEμC** *formula* $\mu Z.\big((\neg\mathsf{pun} \to \mathsf{vote_i}) \vee \langle c\rangle Z\big)$*, which can be viewed as a "naive" fixpoint translation of the formula* $\langle\!\langle c\rangle\!\rangle_{\mathrm{ir}}\mathbf{F}\big(\neg\mathsf{pun} \to \mathsf{vote_i}\big)$ *from Example 2.1.2.*

Upon computing the fixpoint, we find that the entire set of states St is produced. Consequently, we have $M_{vote}, q_0 \models \mu Z.\big((\neg \mathsf{pun} \rightarrow \mathsf{vote_i}) \vee \langle c \rangle Z\big)$*, indicating that the formula is satisfied at state* q_0 *in model* M_{vote}*.*

Proposition 2.1.17 (Bulling and Jamroga (2011)) *The complexity of model checking* **sAEμC** *with strategic modalities* $\langle\langle A \rangle\rangle$ *for* $|A| \leq 2$ *is* **P***-complete and can be performed in time* $O(|\sim| \cdot |\varphi|)$*, where* $|\sim|$ *is the size of the largest equivalence class among* $\sim_1, \ldots, \sim_k$*, and* $|\varphi|$ *is the length of the formula. For* $|A| \geq 3$*, the problem is between* **NP** *and* $\mathbf{\Delta_2^P}$ *with respect to* $|\sim|$ *and* $|\varphi|$*.*

This result indicates that simple alternating epistemic μ-calculus can be an attractive alternative to $\mathbf{ATL}_{\mathrm{ir}}$ from a complexity perspective. However, formulae of $\mathbf{ATL}_{\mathrm{ir}}$ do not admit universal translations to **sAEμC**.

Proposition 2.1.18 (Bulling and Jamroga (2011)) $\mathbf{ATL}_{\mathrm{ir}}$ *and* **sAEμC** *have incomparable expressive power and incomparable distinguishing power.*

The proof that **sAEμC** does not cover the expressive power of $\mathbf{ATL}_{\mathrm{ir}}$ relies on formulae of the form $\langle\langle a \rangle\rangle \mathbf{F}\mathsf{p}$, but a similar argument can be constructed for $\langle\langle a \rangle\rangle \mathbf{G}\mathsf{p}$. Consequently, long-term strategic modalities of $\mathbf{ATL}_{\mathrm{ir}}$ do not have simple fixpoint characterizations in terms of the next-step strategic modalities $\langle A \rangle$.

A similar result was established in (Dima et al., 2014, Theorem 11) for $\mathbf{ATL}_{\mathrm{iR}}$, the variant of **ATL** with imperfect information and perfect recall strategies. Specifically, it was shown that $\langle\langle a \rangle\rangle \mathbf{F}\mathsf{p}$ cannot be expressed as a formula of epistemic μ-calculus under these assumptions. Further results Dima et al. (2014, 2015) confirm that even richer variants of μ-calculus do not cover the full expressive power of $\mathbf{ATL}_{\mathrm{iR}}$.

2.1.7 Natural Strategies

To reason about natural strategic ability, the logic **NatATL** was introduced in Jamroga et al. (2017b, 2019b) with the following syntax:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A \rangle\rangle^{\leq k} \mathbf{X}\varphi \mid \langle\langle A \rangle\rangle^{\leq k} \mathbf{F}\varphi \mid \langle\langle A \rangle\rangle^{\leq k} \mathbf{G}\varphi \mid \langle\langle A \rangle\rangle^{\leq k} \varphi \,\mathbf{U}\, \varphi.$$

Here, A is a group of agents and $k \in \mathbb{N}$ is a complexity bound. Intuitively, the formula $\langle\langle A \rangle\rangle^{\leq k} \gamma$ can be read as "coalition A has a collective strategy of size less than or equal to k to enforce the property γ."

The formulas of **NatATL** utilize classical temporal operators, including:

- **X** (in the next state)
- **G** (always from now on)
- **F** (now or sometime in the future)
- **U** (strong until)

For instance, the formula $\langle\langle cust \rangle\rangle^{\leq 10}\mathbf{F}\mathsf{ticket}$ expresses that the customer can obtain a ticket by a strategy of complexity at most 10. This seems more suitable as a functionality requirement than requiring the existence of any function from states to actions.

It is worth noting that the path quantifier "for all paths" from temporal logic can be defined as $\mathsf{A}\gamma \equiv \langle\langle\emptyset\rangle\rangle^{\leq 0}\gamma$.

2.1.8 Variants and Complexities of ATL

There are several variants of **ATL**, each extending the basic logic in different ways. The complexity of model-checking under these logics can vary significantly. In this section, we examine the most notable extensions and their respective complexities.

ATL* allows for arbitrary nesting of path quantifiers and temporal operators, similar to the extension from **CTL** to **CTL***. The model-checking problem for **ATL*** is 2**EXPTIME**-complete Alur et al. (1998). This increased complexity arises from the need to evaluate deeply nested strategic and temporal properties.

ATLK extends **ATL** by adding knowledge operators, enabling reasoning about the knowledge of agents. This extension is particularly useful in multi-agent systems where understanding the knowledge and beliefs of agents is crucial. The complexity stems from the interplay between strategic reasoning and epistemic considerations and can vary significantly:

- **Perfect Information and Perfect Recall:** Under these conditions, the problem is **P**-complete van der Hoek and Wooldridge (2003), meaning it can be solved efficiently.
- **Imperfect Information and Imperfect Recall:** In scenarios where agents have imperfect information and do not recall past events perfectly, the problem becomes more complex and is $\mathbf{\Delta_2^P}$-complete Jamroga and Ågotnes (2007). This

higher complexity reflects the challenges of reasoning under uncertainty and limited memory.

ATLC introduces communication operators into **ATL**, allowing the expression of scenarios where agents can communicate with each other. This extension is particularly relevant in distributed systems and communication networks. The model-checking complexity for **ATLC** is highly dependent on the specifics of the communication model and the assumptions about message passing and synchronization.

ATLR is another notable variant that incorporates reward operators, enabling reasoning about quantitative aspects such as costs, rewards, or utilities associated with strategies. This extension is useful in economic and resource management applications. The complexity of model-checking **ATLR** can range from **PSPACE**-complete to **EXPSPACE**-complete, depending on the expressiveness of the reward structure and the interaction with strategic reasoning.

These variants demonstrate the richness and versatility of **ATL** in modeling complex systems with various dimensions of reasoning. Understanding the complexity associated with each variant is crucial for selecting the appropriate logic for a given application and for developing efficient model-checking algorithms.

2.2 Available Tools and Approaches

The problem of verification of $\mathbf{ATL}_{\mathrm{Ir}}$ and $\mathbf{ATL}_{\mathrm{ir}}$ formulas has been a long-standing research topic. As a result, multiple model-checking tools have been developed over the years. Each tool implements a different approach to solving the problem, employs distinct data structures, has its own input specification language, and even represents the model in various ways. In this section, we will introduce a selected and well-known subset of the available tools, starting with command-line software and then moving to more user-friendly graphical interfaces. This overview will provide a glimpse into the diverse range of tools available for model checking $\mathbf{ATL}_{\mathrm{Ir}}$ and $\mathbf{ATL}_{\mathrm{ir}}$ formulas.

2.2.1 Command-Line Verifiers

Command-line tools have several advantages over graphical software. Firstly, the textual interface is relatively easy to implement, depending on the chosen programming

language. Additionally, it is typically straightforward to automate tests and measure the efficiency of the implementation using command-line tools.

However, there are also some drawbacks to using command-line tools. The textual form of the model specification can make it challenging to ensure that the model being designed is accurately represented in the input. Without a visual representation of the created states and transitions, it can be difficult to verify that the model matches the intended design.

Similarly, interpreting the verification results can also be more complicated without a graphical representation. It is harder to confirm the correctness of the results, as the lack of visualization makes it more difficult to understand the relationships between different components of the model.

Overall, while command-line tools offer some advantages, they also present some challenges, particularly when it comes to ensuring the accuracy of the model and interpreting the verification results.

MCMAS

One of the most well-known and state-of-the-art tools for verifying multi-agent systems is MCMAS Lomuscio et al. (2017). This tool is equipped with verification algorithms for **ATLK**, **CTLK**, and **CTL**. While it can handle imperfect information semantics for **ATL**, it does it in a very inefficient way, as its primary focus is on model checking under perfect information.

MCMAS is a symbolic model checker, which means that the model is internally represented using an Ordered Binary Decision Diagram (OBDD) structure. This representation can significantly impact the performance of the verification procedures. Although it is possible to generate larger models and store them in memory, the OBDD representation can strongly affect computation times. Both generating the symbolic model and verifying the formula heavily depend on the underlying structure, which in turn depends on the state coding.

To illustrate this, consider the Simple Voting Model shown in Figure 2.1. The ISPL specifications for the Voter and Coercer agents are presented in Figures 2.4 and 2.5, respectively. These specifications demonstrate how the agents' behavior can be formalized using the ISPL language, which is used as input for the MCMAS tool.

MCMAS-SLK

```
Agent Voter1
        Lobsvars = {voter1Voted, voter1Pun, voter1Decided};
        Vars:
                vote: 0..2;
                decision: {give, ng, None};
        end Vars
        Actions = {Vote1, Vote2, Give, Ng, Wait};
        Protocol:
                vote=0: {Vote1, Vote2, Wait};
                vote>0 and decision=None: {Give, Ng, Wait};
                Other: {Wait};
        end Protocol
        Evolution:
                vote=1 if Action=Vote1;
                vote=2 if Action=Vote2;
                decision=give if Action=Give;
                decision=ng if Action=Ng;
        end Evolution
end Agent
```

Figure 2.4: ISPL specification of the Voter in Simple voting model.

A significant enhancement to the MCMAS tool is the development of MCMAS-SLK, a model-checker designed to facilitate the verification of systems against specifications articulated in a specialized variant of Strategy Logic that incorporates epistemic modalities. Building upon the foundation established by MCMAS, MCMAS-SLK adopts a similar symbolic model-checking approach, leveraging the established algorithms from its predecessor for the verification of epistemic specifications.

However, MCMAS-SLK distinguishes itself by implementing innovative labeling algorithms specifically tailored for Strategy Logic. This represents a notable advancement in the tool's capabilities, as it enables the verification of strategic properties in multi-agent systems. By incorporating these specialized algorithms, MCMAS-SLK provides a powerful tool for verifying complex systems against a wide range of specifications, including those that involve strategic and epistemic reasoning.

```
Agent Coercer
        Lobsvars = {voter1Voted, voter1Decided, voter1Pun, voter2Voted, voter2Decided, voter2Pun};
        Vars:
                x: boolean;
        end Vars
        Actions = {Pun1, Np1, Pun2, Np2, Wait};
        Protocol:
                Environment.voter1Voted=true and Environment.voter1Pun=None
                        and Environment.voter1Decided=true: {Pun1, Np1, Wait};
                Environment.voter2Voted=true and Environment.voter2Pun=None
                        and Environment.voter2Decided=true: {Pun2, Np2, Wait};
                Other: {Wait};
        end Protocol
        Evolution:
                x=false if Action=Wait;
        end Evolution
end Agent
```

Figure 2.5: ISPL specification of the Coercer in Simple voting model.

PRISM

PRISM is a probabilistic model checker Kwiatkowska et al. (2002) that enables the construction and analysis of probabilistic models, including discrete-time Markov chains, probabilistic automata, Markov decision processes, and many others. Notably, one of the extensions of PRISM can also be used for the verification of **rPATL**, which is an extension of **PATL**, a probabilistic version of **ATL**, but only for perfect information strategies.

This extension allows users to reason about probabilistic systems with multiple agents, where the agents' actions are subject to probabilistic uncertainty. By leveraging PRISM's capabilities, users can model and analyze complex systems that involve probabilistic and strategic reasoning, making it a powerful tool for verifying the behavior of probabilistic multi-agent systems.

VERICS

VERICS is a SAT-based model checker for Timed Automata Dembiński et al. (2003) that incorporates state-of-the-art methods for bounded and parametric model checking. The tool is composed of multiple modules and implements bounded algorithms for various temporal logics, including:

- **CTL** (Computation Tree Logic)
- Real-time **CTL** (an extension of **CTL** for timed systems)

- An existential fragment of **CTL** and **CTL*** (which allows for reasoning about the existence of paths in a system)

By leveraging SAT-based techniques, VERICS provides an efficient and scalable solution for model checking timed systems, enabling users to verify complex properties and behaviors in a wide range of applications.

SMC

SMC is a model checker for a subset of $\mathbf{ATL}_{\mathrm{ir}}$ Pilecki et al. (2017), specifically designed to facilitate a uniform strategy synthesis algorithm. This tool enables the verification of multi-agent systems with imperfect information and imperfect recall, and provides a novel approach to synthesizing strategies for achieving specific goals in these systems.

By focusing on a subset of $\mathbf{ATL}_{\mathrm{ir}}$, SMC offers a specialized solution for model checking and strategy synthesis in complex multi-agent systems, where agents have limited knowledge and memory. This tool is particularly useful for applications where agents need to make decisions based on incomplete or uncertain information, and where the system's behavior depends on the interactions between multiple agents.

MCK

MCK is a model checker for the logic of knowledge Gammie and Meyden (2004), specifically designed to verify temporal-epistemic specifications. It was the first symbolic model checker based on Ordered Binary Decision Diagrams (OBDDs) to support this type of specification. The underlying temporal logic used by MCK is **CTL*** (Computation Tree Logic with Star), which provides a powerful framework for expressing complex temporal properties.

One of the key features of MCK is its support for various semantics, including:

- Clock semantics
- Observational semantics
- Perfect recall semantics

However, due to the high computational cost, some of these semantics are only supported in a limited form. This is a common challenge in model checking, as the complexity of the semantics can impact the performance of the model checker.

By providing a symbolic model checking approach based on OBDDs, MCK offers an efficient and scalable solution for verifying temporal-epistemic specifications in complex systems. This makes it a valuable tool for researchers and practitioners working in the field of formal verification and artificial intelligence.

MCTK

MCTK is a model checker for temporal logic of knowledge that builds upon the NuSMV model checker Cimatti et al. (2002). To verify epistemic formulas, MCTK employs a novel encoding approach that exploits two key properties:

- Locality of propositions: This refers to the fact that propositions are typically local to specific agents or components in the system.
- Labelling of transitions: This involves annotating the transitions in the system with relevant information, such as the agents involved or the actions taken.

By leveraging these properties, MCTK is able to efficiently encode epistemic formulas and verify them using the underlying NuSMV model checker. This approach enables MCTK to scale to larger systems and handle complex epistemic specifications, making it a valuable tool for researchers and practitioners working in the field of formal verification and artificial intelligence.

Tamarin

Although Tamarin is not formally a model checker, it is a security protocol verification tool Meier et al. (2013) that can be used to verify multi-agent systems, particularly those that involve a large number of cryptographic operations. In Tamarin, the model is specified as a multiset rewriting system, which allows for the analysis of complex systems with multiple interacting components.

The verification process in Tamarin involves checking temporal first-order properties and a message theory, making it well-suited for verifying the cryptographic aspects of e-voting protocols. However, as demonstrated in Bruni et al. (2017); **?**, Tamarin can also be used to verify other properties, such as receipt-freeness and vote-privacy, which are essential for ensuring the integrity and confidentiality of e-voting systems.

While Tamarin's primary focus is on security protocol verification, its capabilities make it a valuable tool for researchers and practitioners working on the verification of multi-agent systems, especially those that involve cryptographic operations and complex interactions between agents.

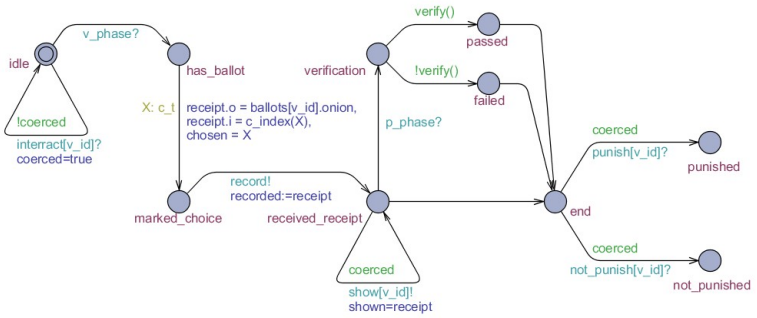

Figure 2.6: Voter template in UPPAAL

2.2.2 Model Checkers with GUI

While research on model checking often focuses on the theoretical aspects of logical systems and verification algorithms, it is essential to remember that a model checking framework is truly valuable when it is used to analyze real-world systems. The analysis does not necessarily need to result in a "correctness certificate" for the system; even a readable model of the system and an understandable formula capturing the requirements can be of significant value.

In this context, two key features of a model checker are crucial. Firstly, it should provide a flexible model specification language that enables modular and succinct specification of processes. This allows users to easily model complex systems and modify their specifications as needed. Secondly, the model checker should offer a good graphical user interface (GUI) that facilitates user interaction and visualization of the model.

Surprisingly, tools that satisfy both criteria are scarce. However, the state-of-the-art **CTL** model checker UPPAAL can provide a suitable environment for modeling and preliminary verification of multi-agent systems, such as voting protocols and their social context, including (to some extent) human and social factors. In this section, we will briefly introduce the modeling side of UPPAAL, highlighting its capabilities and features that make it an attractive choice for modeling and analyzing complex systems.

Modelling in UPPAAL

In UPPAAL, a model is composed of a set of concurrent processes, also referred to as extended timed automata. The semantics of the model define a transition system as a network of these concurrent processes.

Each process in the model is defined by a template, which can have a set of parameters. The use of parameterized templates allows for the definition of multiple almost identical processes, making it easier to model complex systems with many similar components. When a template has one or more parameters, it gives rise to a set of processes, with one process created for each possible combination of parameter values.

A template in Uppaal consists of three main components:

- *Nodes*: These represent the states of the process.
- *Edges*: These represent the transitions between states.
- Optional local declarations: These can be used to define local variables and functions that are specific to the process.

An example of a template is shown in Figure 2.6, which models the behavior of a voter. This template defines the states and transitions that describe the actions of a voter in a voting system, and can be used to create multiple voter processes with similar behavior.

In Uppaal, nodes are represented by circles and correspond to the local states of a module. There are two special types of nodes:

- *Initial* nodes: These are marked by a double circle and represent the starting state of a process.
- *Committed* nodes: These are marked by a circled C and are used to create atomic sequences or encode synchronization between more than two components.

When a process is in a committed node, the next transition must involve an edge from one of the committed nodes. This means that the process is "committed" to taking a specific action or sequence of actions, and cannot be interrupted or preempted by other processes.

Committed nodes are a powerful feature in Uppaal, as they allow modelers to specify complex synchronization behaviors and ensure that certain actions are executed atomically. By using committed nodes, modelers can create more realistic and accurate models of concurrent systems, and verify their behavior using Uppaal's model checking capabilities.

In UPPAAL, edges define the local transitions in a module, specifying the possible next states that a process can move to. Edges are annotated with four types of labels, each with a specific purpose:

- *Selections* (yellow): These specify the possible values that can be chosen for a variable or expression.
- *Guards* (green): These are conditions that must be true for the transition to be taken.
- *Synchronizations* (teal): These specify the synchronization actions that must be performed with other processes.
- *Updates* (blue): These specify the updates to be made to variables or expressions as a result of the transition.

The syntax of expressions in UPPAAL is similar to that of C/C++, with a few key differences:

- Boolean values are evaluated as integers 0 or 1.
- Quantifiers 'forall(id:type) expr;' and 'exists(id:type) expr;' evaluate to a Boolean value, allowing for more expressive and concise specifications.
- Variable references (pointers) are used differently in UPPAAL, providing a way to access and manipulate variables in a more flexible and efficient manner.

These features allow modelers to specify complex behaviors and interactions between processes in a concise and expressive way, making it easier to model and verify concurrent systems using UPPAAL.

In UPPAAL, the four types of labels on edges have the following meanings:

- *Selections*: These bind an identifier to a value from a given range in a nondeterministic way. This allows for modeling of uncertain or arbitrary choices.
- *Guards*: These enable the transition if and only if the guard condition evaluates to true. This ensures that the transition is only taken when certain conditions are met.

- *Synchronizations*: These allow processes to synchronize over a common channel, labeled `ch?` in the receiver process and `ch!` for the sender. A transition on the sender side can only be fired if there exists an enabled transition on the receiving side labeled with the same channel identifier, and vice versa. This ensures that the sender and receiver are synchronized and can communicate with each other.

- *Update* expressions: These are evaluated when the transition is taken. For a synchronizing transition, the update expressions on the sender side are executed before the receiver ones. This allows for modeling of the effects of the synchronization on the system state.

It is worth noting that straightforward value passing over a channel is not allowed in Uppaal. Instead, shared global variables must be used for transmission, and a committed node is used to ensure that the update expressions are executed in the correct order.

For convenience, in Figure 2.6, the selections and guards are placed at the top or left of an edge, and the synchronizations and updates are placed at the bottom/right. This notation helps to clarify the meaning of the edge labels and makes it easier to read and understand the model.

2.2.3 Challenges and Tradeoffs

When approaching a model-checking problem, it's crucial to carefully select the right tool for the task. As discussed in this section, each tool has its own strengths and weaknesses, and each one approaches the problem from a different perspective. However, there are other important considerations to keep in mind before diving into the tool documentation.

One key aspect is to ensure that the model being created accurately represents the intended system or concept. Tools with good user experience and graphical interfaces, such as Uppaal, can be incredibly helpful in this regard. They allow users to visually design and interact with their models, making it easier to identify potential issues or inconsistencies.

On the other hand, tools with textual input may offer more advanced verification methods, especially for $\mathbf{ATL}_{\mathrm{ir}}$. However, designing the model specification can be a

tedious and error-prone process. This is where combining multiple tools can be beneficial. For example, UPPAAL can be used to design the model using its graphical user interface, and then another tool can be used to run the verification process.

To achieve this, a conversion tool would be necessary to convert the UPPAAL model specification into a format compatible with the other tool. While this may require additional effort, it can be a worthwhile investment for solving more complex problems.

Ultimately, there is no perfect tool that meets all needs. Instead, the best approach may be to combine multiple tools to leverage their respective strengths and overcome their limitations. By doing so, users can create a more comprehensive and effective model-checking workflow.

Part I

New Methods and Algorithms for Verification of Strategic Ability

3

Model Checking ATL_{ir} by Fixpoint Approximation

The problem of verifying strategic abilities under imperfect information is complex, and one of the main reasons for this complexity is that fixpoint equivalences do not hold in $\mathbf{ATL}_{\mathrm{ir}}$, unlike in the perfect information case. This means that we need to explore alternative approaches to reduce the complexity bounds, at least for certain scenarios. In this chapter, we investigate one such approach.

The main idea is that, instead of performing exact model checking, it may be sufficient to provide lower and upper bounds for the output. Specifically, given a formula φ, we aim to construct two translations, $tr_L(\varphi)$ and $tr_U(\varphi)$, such that $tr_L(\varphi) \Rightarrow \varphi \Rightarrow tr_U(\varphi)$. If $tr_L(\varphi)$ is verified as true, then the original formula φ must also hold in the given model. Conversely, if $tr_U(\varphi)$ evaluates to false, then φ must also be false.

For this approach to be useful, the truth values for the translations should be easier to compute than for the original formula. To achieve this, we will build our approximations of $\langle\langle A \rangle\rangle_{\mathrm{ir}}$ on fixpoint-definable properties, mapping the formulae of $\mathbf{ATL}_{\mathrm{ir}}$ to an appropriate variant of alternating μ-calculus.

Throughout this chapter, we use the following notation:

- φ, ψ denote arbitrary formulae of $\mathbf{ATL}_{\mathrm{ir}}$.
- γ denotes arbitrary path subformulae of $\mathbf{ATL}_{\mathrm{ir}}$.
- M is an iCGS (imperfect information concurrent game structure).

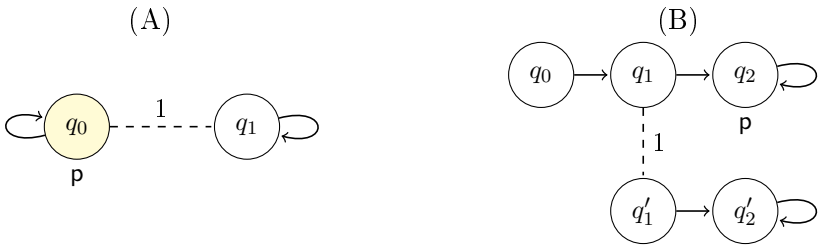

Figure 3.1: Counterexamples for tr_{L1}: (A) M_1; (B) M_2

- q is a state in M, unless explicitly stated otherwise.

The content of this chapter is based on the paper Jamroga et al. (2019a).

3.1 Lower and Upper Bound

The complexity of model checking for **sAEμC** is more attractive than that of **ATL**$_{\text{ir}}$ Alur et al. (2002). However, the expressivity results in Section 2.1.6 indicate that it is not possible to translate **ATL**$_{\text{ir}}$ modalities exactly into simple fixpoint formulae based on standard epistemic and short-term strategic operators.

Instead, we aim to find a translation tr_L that provides a lower bound of the actual strategic abilities. This means that if $M, q \models tr_L(\langle\langle A\rangle\rangle_{\text{ir}}\gamma)$, then we can conclude that $M, q \models \langle\langle A\rangle\rangle_{\text{ir}}\gamma$. In other words, the translation can only reduce, but never enhance the abilities of the coalition.

We start by examining the "naive" fixpoint translation that mimics the one for **ATL**$_{\text{Ir}}$, but show that it is not effective. We then propose an alternative approach that modifies the semantics of the next-time modality to obtain a general lower bound.

Our initial focus is on reachability goals, expressed by formulae $\langle\langle A\rangle\rangle_{\text{ir}}\mathbf{F}\varphi$. We then extend our approach to cover other modalities. By doing so, we aim to develop a translation that provides a useful approximation of the strategic abilities of the coalition, even if it is not exact.

3.1.1 Trying it Simple for Reachability Goals

We begin by considering the simplest translation, analogous to the one in Alur et al. (2002):

$$tr_{L1}(\langle\langle A\rangle\rangle_{\text{ir}}\mathbf{F}\varphi) = \mu Z.(\varphi \vee \langle A\rangle Z).$$

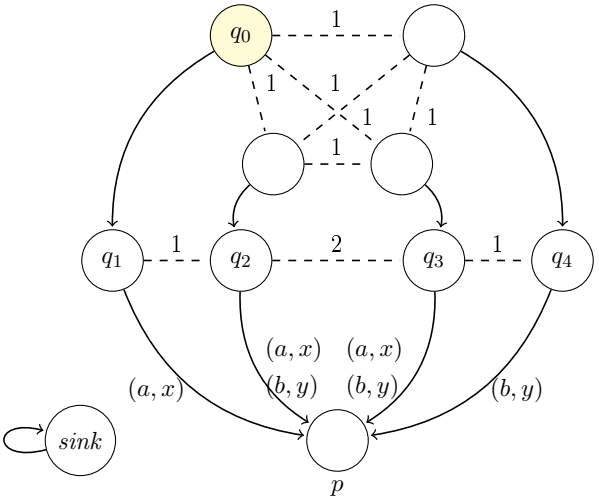

Figure 3.2: M_3: a counterexample for tr_{L2}

However, this translation does not provide a lower or upper bound for the original formula.

To see why, consider model M_1 in Figure 3.1A. We have that $M_1, q_0 \models \mu Z.(\mathsf{p} \vee \langle 1 \rangle Z)$, which follows from the fact that $M, q \models \varphi$ implies $M, q \models \mu Z.(\varphi \vee \langle A \rangle Z)$, for all $A \subseteq \mathbb{A}\mathrm{gt}$. However, $M_1, q_0 \not\models \langle\!\langle 1 \rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\mathsf{p}$, since the only path starting from q_1 loops at the source, never reaching p.

On the other hand, consider model M_2 in Figure 3.1B. We have that $M_2, q_0 \models \langle\!\langle 1 \rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\mathsf{p}$ via the only possible strategy (note that the sequence $q_1' q_2'$ is never visited when starting from q_0). However, $M_2, q_0 \not\models \mu Z.(\mathsf{p} \vee \langle 1 \rangle Z)$, as no strategy can enforce p from $[q_1]_{\sim_1} = \{q_1, q_1'\}$.

As a consequence, we obtain the following result:

Proposition 3.1.1 (Jamroga et al. (2019a)) *$M, q \models \mu Z.(\varphi \vee \langle A \rangle Z)$ does not imply $M, q \models \langle\!\langle A \rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\varphi$. The converse implication does not hold either.*

This proposition shows that the simple translation tr_{L1} does not provide a lower or upper bound for the original formula, and therefore is not suitable for approximating the strategic abilities of the coalition.

Let us now consider a slightly stronger fixpoint specification:

$$tr_{L2}(\langle\!\langle A \rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\varphi) = \mu Z.(E_A \varphi \vee \langle A \rangle Z).$$

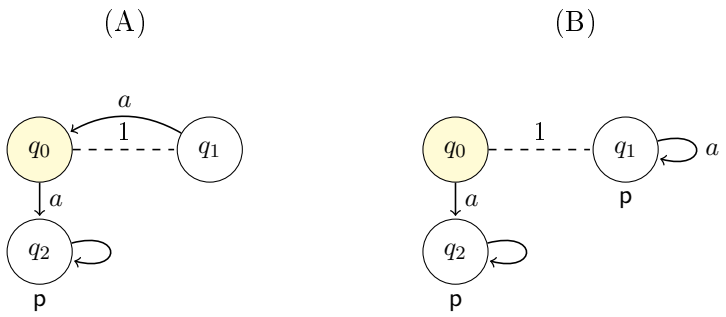

Figure 3.3: Lower bounds are not tight: (A) M_4; (B) M_5

This new translation works for the empty coalition and for single agents, but not for coalitions of multiple players.

We have the following result:

Proposition 3.1.2 (Jamroga et al. (2019a)) *Let $A \subseteq \mathbb{A}\mathrm{gt}$ and $q \in St$. The following holds:*

1. $M, q \models \mu Z.(E_\emptyset \varphi \vee \langle\emptyset\rangle Z)$ *iff* $M, q \models \langle\!\langle\emptyset\rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\varphi$;
2. *If* $|A| = 1$, *then* $M, q \models \mu Z.(E_A \varphi \vee \langle A\rangle Z)$ *implies* $M, q \models \langle\!\langle A\rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\varphi$, *but the converse does not universally hold;*
3. *If* $|A| > 1$, *then* $M, q \models \mu Z.(E_A \varphi \vee \langle A\rangle Z)$ *does not imply* $M, q \models \langle\!\langle A\rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\varphi$. *The converse does not hold either.*

The proof of this proposition can be summarized as follows. The first case follows from known results on games with perfect information. The second case is more involved: a strategy that witnesses $M, q \models \langle\!\langle 1\rangle\!\rangle_{\mathrm{ir}} \mathbf{F}\varphi$ is built step-by-step while computing the fixed point $\mu Z.(E_A\varphi \vee \langle 1\rangle Z)$. The full proof can be found in Jamroga et al. (2019a).

This proposition shows that the translation tr_{L2} works for the empty coalition and for single agents, but not for coalitions of multiple players. This suggests that the translation is not strong enough to capture the strategic abilities of coalitions in general.

According to Propositions 3.1.1 and 3.1.2, the translation tr_{L2} provides lower bounds for $\mathbf{ATL}_{\mathrm{ir}}$ verification only in a limited number of instances. Moreover, the bound is rather loose, as demonstrated by the following example.

Example 3.1.3 *Consider the single-agent iCGS M_4 presented in Figure 3.3A. The only available strategy, in which agent 1 always selects action a, enforces eventually reaching* p*, i.e.,* $M_4, q_0 \models \langle\langle 1 \rangle\rangle_{\text{ir}} \mathbf{F}\mathsf{p}$*. However,* $M_4, q_0 \not\models \mu Z.(K_1\mathsf{p} \vee \langle 1 \rangle Z)$*. This is because the next-step operator in* $\mathbf{ATL}_{\text{ir}}$ *requires reaching* p *simultaneously from all states indistinguishable from* q_0*, whereas* p *is reached from* q_0 *and* q_1 *in one and two steps, respectively.*

This example illustrates that the translation tr_{L2} can be too restrictive, requiring simultaneous reachability from all indistinguishable states. In contrast, the original $\mathbf{ATL}_{\text{ir}}$ formula allows for more flexibility in achieving the goal. This highlights the need for a more refined translation that better captures the strategic abilities of agents in iCGS.

3.1.2 Steadfast Next Step Operator

To obtain a tighter lower bound that works universally, we introduce a new modality $\langle A \rangle^{\bullet}$. This modality can be seen as a semantic variant of the next-step ability operator $\langle A \rangle$, with two key differences:

- Agents in A look for a short-term strategy that succeeds from the "common knowledge" neighborhood of the current state, rather than from the "everybody knows" neighborhood.
- They are allowed to "steadfastly" pursue their goal in a variable number of steps within the indistinguishability class.

To formalize this, we define an auxiliary function *Reach* that collects all states $q \in Q$ such that all paths executing strategy s_A from q eventually reach φ without leaving Q, except possibly for the last step. This is defined as:

$$\begin{aligned} \mathit{Reach}_M(s_A, Q, \varphi) \;=\; \{q \in Q \mid \forall \lambda \in \mathit{out}(q, s_A)\ \exists i\ .\ M, \lambda[i] \models \varphi \\ \text{and } \forall 0 \leq j < i\ .\ \lambda[j] \in Q\}. \end{aligned}$$

In other words, every outcome path must stay in Q until it reaches φ. This function provides a way to capture the idea of agents "steadfastly" pursuing their goal within a certain region of the state space.

We define the *steadfast next-step operator* $\langle A \rangle^{\bullet}$ as follows:

$$M, q \models \langle A \rangle^{\bullet} \varphi \text{ iff there is } s_A \in \Sigma_A^{\text{ir}} \text{ such that } \mathit{Reach}_M(s_A, [q]_{\sim_A^C}, \varphi) = [q]_{\sim_A^C}.$$

In other words, φ must be reached from every state in $[q]_{\sim_A^C}$.

We then propose our final attempt at the lower bound for reachability goals:

$$tr_{L3}(\langle\!\langle A \rangle\!\rangle_{\text{ir}} \mathbf{F} \varphi) = \mu Z.(E_A \varphi \vee \langle A \rangle^{\bullet} Z).$$

This leads to the following result:

Proposition 3.1.4 (Jamroga et al. (2019a)) *If $M, q \models \mu Z.(E_A \varphi \vee \langle A \rangle^{\bullet} Z)$, then $M, q \models \langle\!\langle A \rangle\!\rangle_{\text{ir}} \mathbf{F} \varphi$. The converse does not universally hold.*

The proof of this proposition is similar to the proof of the second case of Proposition 3.1.2. We synthesize a strategy witnessing $M, q \models \langle\!\langle A \rangle\!\rangle_{\text{ir}} \mathbf{F} \varphi$ during the iterative computation of the fixpoint $\mu Z.(E_A \varphi \vee \langle A \rangle^{\bullet} Z)$. The key observation is that the operator $\langle A \rangle^{\bullet} Z$ selects those states for which the entire common knowledge neighborhood for group A is enforced into Z. The full proof can be found in Jamroga et al. (2019a).

Thus, tr_{L3} provides a lower bound for reachability goals expressed in $\mathbf{ATL}_{\text{ir}}$. This means that if $M, q \models \mu Z.(E_A \varphi \vee \langle A \rangle^{\bullet} Z)$, then we can conclude that $M, q \models \langle\!\langle A \rangle\!\rangle_{\text{ir}} \mathbf{F} \varphi$. However, the converse does not universally hold, meaning that there may be cases where $M, q \models \langle\!\langle A \rangle\!\rangle_{\text{ir}} \mathbf{F} \varphi$ but $M, q \not\models \mu Z.(E_A \varphi \vee \langle A \rangle^{\bullet} Z)$.

3.1.3 Lower Bounds for "Always" and "Until"

We now extend the translation and the result in Proposition 3.1.4 to all the modalities of $\mathbf{ATL}_{\text{ir}}$. We define the translation for the always and until operators as follows:

$$\begin{aligned} tr_{L3}(\langle\!\langle A \rangle\!\rangle_{\text{ir}} \mathbf{G} \varphi) &= \nu Z.(C_A \varphi \wedge \langle A \rangle^{\bullet} Z), \\ tr_{L3}(\langle\!\langle A \rangle\!\rangle_{\text{ir}} \psi \, \mathbf{U} \, \varphi) &= \mu Z.\big(E_A \varphi \vee (C_A \psi \wedge \langle A \rangle^{\bullet} Z)\big). \end{aligned}$$

We then obtain the following result:

Theorem 3.1.5 (Jamroga et al. (2019a))

1. *If $M, q \models \nu Z.(C_A \varphi \wedge \langle A \rangle^{\bullet} Z)$, then $M, q \models \langle\!\langle A \rangle\!\rangle_{\text{ir}} \mathbf{G} \varphi$;*
2. *If $M, q \models \mu Z.\big(E_A \varphi \vee (C_A \psi \wedge \langle A \rangle^{\bullet} Z)\big)$, then $M, q \models \langle\!\langle A \rangle\!\rangle_{\text{ir}} \psi \, \mathbf{U} \, \varphi$.*

The proof of this theorem uses techniques similar to the proofs of Propositions 3.1.2 and 3.1.4. The main difference is in the first case, where a strategy witnessing $M, q \models \langle\langle A \rangle\rangle_{\text{ir}} \mathbf{G}\varphi$ is built decrementally while computing the fixpoint $M, q \models \nu Z.(C_A\varphi \wedge \langle A \rangle^\bullet Z)$. The whole proof can be found in Jamroga et al. (2019a).

A closer inspection of the proof reveals that an even stronger result can be obtained:

Remark 3.1.6

- $M, q \models tr_{L3}(\langle\langle A \rangle\rangle_{\text{ir}} \mathbf{G}\varphi)$ *implies* $M, q \models \langle\langle A \rangle\rangle_{\text{ir}} \mathbf{G} C_A\varphi$;
- $M, q \models tr_{L3}(\langle\langle A \rangle\rangle_{\text{ir}} \psi \,\mathbf{U}\, \varphi)$ *implies* $M, q \models \langle\langle A \rangle\rangle_{\text{ir}} (C_A\psi) \,\mathbf{U}\, (E_A\varphi)$.

This remark shows that the translation tr_{L3} provides an even tighter lower bound for the always and until operators.

3.1.4 Discussion & Properties

Theorem 3.1.5 establishes that $tr_{L3}(\varphi)$ provides a correct lower bound for the value of φ for all formulae of $\mathbf{ATL}_{\text{ir}}$. In this section, we examine the tightness of this approximation from a theoretical perspective.

First, in Section 3.1.4.1, we argue that the use of a non-standard next-step ability operator is justified, as it enables us to obtain strictly tighter approximations than the standard one. This suggests that the novel operator is a valuable addition to the framework, allowing for more accurate approximations.

Next, in Section 3.1.4.2, we present a partial characterization of models for which the lower bound is tight. Specifically, we provide a necessary condition (Proposition 3.1.9) that states that if the lower bound verification for $\langle\langle A \rangle\rangle\gamma$ returns "true," then the agents in A must have a *recomputable* strategy s_A to enforce γ. This means that the agents will continue to know that s_A is winning for γ at any point during the execution of s_A.

In other words, the lower bound is tight only if the agents have a strategy that is "recomputable" in the sense that they can continually reassess and confirm that the strategy is winning. This provides insight into the nature of the approximation and highlights the importance of recomputability in achieving tight bounds.

3.1.4.1 Comparing tr_{L2} and tr_{L3} for Reachability Goals

Translation tr_{L3} updates tr_{L2} by replacing the standard next-step ability operator $\langle A\rangle$ with the "steadfast next-step ability" $\langle A\rangle^\bullet$. The difference between the semantics of $\langle A\rangle\varphi$ and $\langle A\rangle^\bullet\varphi$ is twofold.

First, $\langle A\rangle\varphi$ looks for a winning short-term strategy in the "everybody knows" neighborhood of a given state (i.e., $[q]_{\sim_A^E}$), whereas $\langle A\rangle^\bullet\varphi$ looks at the "common knowledge" neighborhood (i.e., $[q]_{\sim_A^C}$).

Secondly, $\langle A\rangle^\bullet$ allows to "zig-zag" across $[q]_{\sim_A^C}$ until a state satisfying φ is found.

Actually, the first change would suffice to provide a universally correct lower bound for **ATL**$_{\text{ir}}$. The second update makes it more useful in models where agents may not see the occurrence of some action, such as M_4 of Figure 3.3A.

To see this formally, we show that tr_{L3} provides a strictly tighter approximation than tr_{L2} on singleton coalitions:

Proposition 3.1.7 (Jamroga et al. (2019a)) *If $M, q \models \mu Z.(K_a\varphi \vee \langle a\rangle Z)$, then $M, q \models \mu Z.(K_a\varphi \vee \langle a\rangle^\bullet Z)$. The converse does not universally hold.*

The proof can be found in Jamroga et al. (2019a).

This proposition shows that tr_{L3} provides a strictly tighter approximation than tr_{L2} for singleton coalitions, making it a more useful and accurate tool for model checking **ATL**$_{\text{ir}}$.

On the other hand, if agent a always sees whenever an action occurs, then tr_{L2} and tr_{L3} coincide for a's abilities. To formalize this, we introduce the concept of an iCGS being "lockstep for a". An iCGS M is lockstep for a if, whenever there is a transition from q to q' in M, we have $q \not\sim_a q'$.

The following proposition is straightforward:

Proposition 3.1.8 (Jamroga et al. (2019a)) *If M is lockstep for a, then $M, q \models \langle a\rangle\varphi$ iff $M, q \models \langle a\rangle^\bullet\varphi$. In consequence, $M, q \models tr_{L2}(\langle\!\langle a\rangle\!\rangle \mathbf{F}\varphi)$ iff $M, q \models tr_{L3}(\langle\!\langle a\rangle\!\rangle \mathbf{F}\varphi)$.*

This proposition shows that if the iCGS is lockstep for agent a, then the two translations tr_{L2} and tr_{L3} are equivalent for a's abilities. This means that in such cases, using either translation will yield the same result, and tr_{L2} may be sufficient for model checking a's abilities.

3.1.4.2 When is the Lower Bound Tight?

An interesting question is: what is the subclass of iCGS's for which tr_{L3} is tight, i.e., the answer given by the approximation is exact? We address this question only partially here. In fact, we characterize a subclass of iCGS's for which tr_{L3} is certainly not tight, by the necessary condition below.

Let $\gamma \equiv \mathbf{G}\psi$ or $\gamma \equiv \psi_1 \mathbf{U} \psi_2$ for some $\psi, \psi_1, \psi_2 \in \mathbf{ATL}_{\mathrm{ir}}$. We say that strategy $s_A \in \Sigma_A^{\mathrm{ir}}$ is winning for γ from q if it obtains γ for all paths in $out^{\mathrm{ir}}(q, s_A)$. Moreover, for such s_A, let $RR(q, s_A, \gamma)$ be the set of relevant reachable states of s_A in the context of γ, defined as follows:

- $RR(q, s_A, \mathbf{G}\psi)$ is the set of states that occur anywhere in $out^{\mathrm{ir}}(q, s_A)$.
- $RR(q, s_A, \psi_1 \mathbf{U} \psi_2)$ is the set of states that occur anywhere in $out^{\mathrm{ir}}(q, s_A)$ before the first occurrence of ψ_2.

We have the following result:

Proposition 3.1.9 (Jamroga et al. (2019a)) *Let M be an iCGS, $q \in St_M$. Furthermore, suppose that $M, q \models tr_{L3}(\langle\langle A \rangle\rangle_{\mathrm{ir}} \gamma)$, i.e., the lower bound translation of $\langle\langle A \rangle\rangle_{\mathrm{ir}} \gamma$ returns true in M, q. Then, there is a strategy $s_A \in \Sigma_A^{\mathrm{ir}}$ which is winning for γ from every $q' \in RR(q, s_A, \gamma)$.*

The proof is straightforward from the fact that $tr_{L3}(\varphi)$ is a fixpoint formula, and model checking of $tr_{L3}(\varphi)$ produces a winning strategy for γ from q.

Conversely, the approximation is not tight if there are winning strategies, but each of them reaches an intermediate state q' from which no winning follow-up strategy can be computed. This can only happen if some states in the epistemic neighborhood $[q']_{\sim_A^E}$ are not reachable by s_A. In consequence, the agents in A forget relevant information that comes solely from the fact that they are executing s_A.

In other words, the approximation is not tight if the agents in A lose information about their strategy during its execution, which prevents them from computing a winning follow-up strategy. This highlights the importance of considering the epistemic neighborhoods of states in the model checking process.

3.1.5 Upper Bound

In a given model, it is always the case that $\Sigma_A^{\text{ir}} \subseteq \Sigma_A^{\text{Ir}}$. This means that whatever coalition A can achieve according to the ir-semantics, they can also achieve it according to the Ir-semantics.

Conversely, if the agents have no perfect information strategy to achieve γ, they cannot have an imperfect information strategy to obtain the same. This observation allows us to define a simple upper bound for formulae of $\mathbf{ATL}_{\text{ir}}$:

Proposition 3.1.10 (Jamroga et al. (2019a)) *Let M be an iCGS and $q \in St_M$ a state in M. Then, $M, q \models \langle\langle A \rangle\rangle_{\text{ir}} \gamma$ implies $M, q \models E_A \langle\langle A \rangle\rangle_{\text{Ir}} \gamma$.*

The proof of this proposition is straightforward from the semantics.

This proposition provides an upper bound for formulae of $\mathbf{ATL}_{\text{ir}}$, which can be used to approximate the truth value of a given formula. Specifically, if $M, q \not\models E_A \langle\langle A \rangle\rangle_{\text{Ir}} \gamma$, then we can conclude that $M, q \not\models \langle\langle A \rangle\rangle_{\text{ir}} \gamma$. This can be useful in practice, as it allows us to quickly eliminate some possibilities and focus on the remaining ones.

3.1.6 Approximation Semantics for $\mathbf{ATL}_{\text{ir}}$

Based on Theorem 3.1.5 and Proposition 3.1.10, we propose the lower approximation tr_L and the upper approximation tr_U for all formulae of $\mathbf{ATL}_{\text{ir}}$ as follows:

$$
\begin{aligned}
tr_L(p) &= p,\\
tr_L(\neg\varphi) &= \neg tr_U(\varphi),\\
tr_L(\varphi \wedge \psi) &= tr_L(\varphi) \wedge tr_L(\psi),\\
tr_L(\langle A\rangle\varphi) &= \langle A\rangle tr_L(\varphi),\\
tr_L(\langle\!\langle A\rangle\!\rangle_{\text{ir}}\mathbf{G}\varphi) &= \nu Z.(C_A tr_L(\varphi) \wedge \langle A\rangle^{\bullet} Z),\\
tr_L(\langle\!\langle A\rangle\!\rangle_{\text{ir}}\psi\,\mathbf{U}\,\varphi) &= \mu Z.\big(E_A tr_L(\varphi) \vee (C_A tr_L(\psi) \wedge \langle A\rangle^{\bullet} Z)\big).
\end{aligned}
$$

$$
\begin{aligned}
tr_U(p) &= p,\\
tr_U(\neg\varphi) &= \neg tr_L(\varphi),\\
tr_U(\varphi \wedge \psi) &= tr_U(\varphi) \wedge tr_U(\psi),\\
tr_U(\langle A\rangle\varphi) &= E_A\langle\!\langle A\rangle\!\rangle_{\text{Ir}}\mathbf{X}tr_U(\varphi),\\
tr_U(\langle\!\langle A\rangle\!\rangle_{\text{ir}}\mathbf{G}\varphi) &= E_A\langle\!\langle A\rangle\!\rangle_{\text{Ir}}\mathbf{G}tr_U(\varphi),\\
tr_U(\langle\!\langle A\rangle\!\rangle_{\text{ir}}\psi\,\mathbf{U}\,\varphi) &= E_A\langle\!\langle A\rangle\!\rangle_{\text{Ir}} tr_U(\psi)\,\mathbf{U}\,tr_U(\varphi).
\end{aligned}
$$

Note that in computing the upper approximation, every application of rules for $tr_L(\langle\!\langle A\rangle\!\rangle_{\text{ir}}\mathbf{G}\varphi)$ and $tr_L(\langle\!\langle A\rangle\!\rangle_{\text{ir}}\psi\,\mathbf{U}\,\varphi)$ selects a fresh variable to be bound by ν and μ, respectively.

We then obtain the following result:

Theorem 3.1.11 (Jamroga et al. (2019a)) *For any iCGS M, state q in it, and* $\mathbf{ATL}_{\text{ir}}$ *formula φ:*

$$M, q \models tr_L(\varphi) \quad\Rightarrow\quad M, q \models \varphi \quad\Rightarrow\quad M, q \models tr_U(\varphi).$$

The proof of this theorem is a straightforward induction on the structure of φ.

We also obtain the following result regarding the complexity of model checking the approximations:

Theorem 3.1.12 (Jamroga et al. (2019a)) *If φ includes only coalitions of size at most 1, then model checking $tr_L(\varphi)$ and $tr_U(\varphi)$ can be done in time $O(|M|\cdot|\varphi|)$. In the general case, the problem of model checking $tr_L(\varphi)$ and $tr_U(\varphi)$ is between* **NP** *and* $\mathbf{\Delta_2^P}$ *wrt* $\max_{A\in\varphi}(|\sim_A^C|)$ *and* $|\varphi|$.

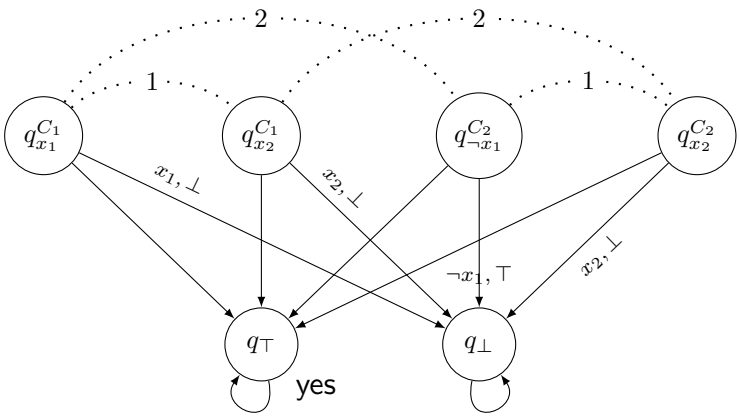

Figure 3.4: Model M_Φ for $\Phi \equiv C_1 \wedge C_2, C_1 \equiv x_1 \vee x_2, C_2 \equiv \neg x_1 \vee x_2$. Only transitions leading to $q_\bot$ are labeled; the other combinations of actions lead to $q_\top$.

The idea of the proof of this theorem is as follows. Firstly, note that both translations work in linear time and produce formulae of size linear in the size of the original. The proof follows by induction on the structure of φ. The interesting cases are $tr_U(\langle\langle A\rangle\rangle_{\text{ir}}\psi)$ and $tr_L(\langle\langle A\rangle\rangle_{\text{ir}}\psi)$, where ψ contains no strategic modalities. For the former, recall that model checking of $\langle\langle A\rangle\rangle_{\text{Ir}}\psi$ is in **P** wrt to the size of the model and the formula Alur et al. (2002). For the latter, it suffices to determine the model checking complexity for formulae $\langle A\rangle^\bullet\psi$, where ψ contains no strategic modalities. The whole proof can be found in Jamroga et al. (2019a).

This theorem shows that model checking the approximations can be done efficiently in certain cases, and provides a bound on the complexity of model checking in the general case.

Our approximations offer a computational advantage over exact **ATL**$_{\text{ir}}$ model checking in cases where the common knowledge neighborhoods for coalition A are significantly smaller than the whole model. This occurs when the members of A have similar knowledge, and especially when the coalition consists of a single agent.

It is worth noting that this result is analogous to several existing results for solving multi-player games with imperfect information and perfect recall. Specifically:

- Verifying abilities of individual agents is decidable, even though the general problem is undecidable Chatterjee et al. (2007).

- Coalitions with exactly the same knowledge can be verified more efficiently Guelev et al. (2011).
- Hierarchical knowledge structures can be exploited to improve verification efficiency Berwanger and Kaiser (2010); Berwanger et al. (2015).
- Knowledge coming from publicly visible actions can also be used to improve verification efficiency Belardinelli et al. (2017b).

These results suggest that there are certain structural properties of the model that can be exploited to improve the efficiency of verification, and our approximations for $\mathbf{ATL}_{\mathrm{ir}}$ model checking are another example of this phenomenon.

3.1.7 Approximation of Abilities under Perfect Recall

Our current analysis primarily investigates strategic ability approximation under memoryless strategies in $\mathbf{ATL}_{\mathrm{ir}}$. While such approximations could theoretically extend to $\mathbf{ATL}_{\mathrm{iR}}$ (the variant employing uniform perfect recall strategies), the inherent computational intractability of $\mathbf{ATL}_{\mathrm{iR}}$ model checking Alur et al. (2002) indicates that significant methodological modifications would be required to achieve comparable approximation quality.

Notwithstanding these challenges, we establish two critical theoretical connections:

- Any coalition possessing a successful memoryless strategy necessarily admits a winning perfect recall strategy. This implies that our lower-bound guarantees for $\mathbf{ATL}_{\mathrm{ir}}$ remain valid under $\mathbf{ATL}_{\mathrm{iR}}$ semantics.
- Existing literature confirms the semantic equivalence between $\mathbf{ATL}_{\mathrm{IR}}$ and $\mathbf{ATL}_{\mathrm{Ir}}$ Schobbens (2004), ensuring that our upper-bound analyses also hold for $\mathbf{ATL}_{\mathrm{iR}}$.

These dual observations enable practical application of our framework to perfect recall scenarios. We observe that many benchmark models implement perfect recall through state representations explicitly encoding agents' observational history. In such cases, the distinction between memoryless and perfect recall semantics collapses entirely. Our experimental validation demonstrates that the proposed approximation techniques

achieve significant performance improvements in these perfect recall models while preserving semantic equivalence. This finding suggests that memoryless strategy approximations can effectively support verification tasks traditionally requiring perfect recall semantics, provided the state representation maintains complete observational history.

3.2 Optimization of Data Structures and Operations

The baseline fixpoint algorithms for computing strategic ability bounds admit multiple optimization opportunities. We identify three orthogonal enhancement dimensions that improve computational efficiency while maintaining semantic integrity.

1. Epistemic State Aggregation: The lower bound computation can be reformulated to operate on epistemic equivalence classes rather than individual states. This state space partitioning strategy enables transition relation compression by representing connections between observational equivalence classes instead of atomic states. The resulting state aggregation reduces both memory footprint and computational complexity by eliminating redundant calculations across indistinguishable states.

2. Disjoint-Set Data Structure Optimization: We introduce the Union-Find (disjoint-set) data structure to accelerate set operations during bound computation. By employing path compression and union-by-rank heuristics, this approach reduces time complexity for union and membership operations from linear to near-constant order. This optimization particularly benefits iterative fixpoint computations where dynamic set management dominates execution time.

3. Tree-Structured iCGS Memory Compression: For verification tasks involving the bridge model's tree-like interpreted system structure, we exploit its stratified dependencies. Since each stratum only depends on its immediate predecessor, we implement a layer-wise computation scheme that discards obsolete state information after each transition layer. This incremental processing strategy reduces memory consumption asymptotically while preserving full observational history through structural encoding in state representations.

The first two optimizations are model-independent, and applicable to arbitrary concurrent game structures. The third strategy, while model-specific, reveals broader relevance: its stratification-based compression generalizes to any system exhibiting acyclic,

hierarchical dependencies. Our experiments confirm these optimizations achieve orders-of-magnitude improvements in both memory usage and execution speed, particularly in large-scale perfect recall models where state space complexity would otherwise prove prohibitive.

3.2.1 Reduction Based on Epistemic Equivalence Classes

Consider the fixpoint algorithm computing the lower bound of $M, q \models \langle\langle A \rangle\rangle_{\text{ir}} \gamma$. The crucial part of the algorithm is the computation of the pre-image of the current set of "candidate" states Q. That is, the algorithm looks for the states q' such that $M, q' \models \langle A \rangle^{\bullet} Q$. We observe now that this property is invariant with respect to the common knowledge relation $\sim_A^C$. In other words, either all the states in the common knowledge neighborhood $[q']_{\sim_A^C}$ are in the pre-image, or none of them. In consequence, when searching for one-step strategies, it suffices to consider outgoing transitions per equivalence class of $\sim_A^C$, and not for each global state separately.

Our first optimization consists of an abstraction of the transition space to transitions between equivalence classes of $\sim_A^C$, and an abstraction of the action space by available one-step strategies of the coalition. This means that we keep the information about individual states only to produce the epistemic classes and transitions between them. After the model has been generated, all the relevant information is "moved" to the representative of the class. In particular, the outgoing transitions from any state in class $[q']_{\sim_A^C}$ are moved to its representative.

The epistemic abstraction allows to obtain a significant speedup of the lower bound computation, especially for singleton coalitions. This is because we "pre-compile" the explicit model in a way that reduces the complexity of checking whether all the states in $[q']$ are in the pre-image of Q. The complexity decreases from quadratic with respect to the size of $[q']$ in the naive implementation, to constant in the optimized one.

3.2.2 Optimization of Data Structures Based on Disjoint Sets

The fixpoint algorithms computing $\mathbf{ATL}_{\text{ir}}$'s lower/upper bounds exhibit critical dependency on efficient set operations over epistemic equivalence classes. Given that these sets emerge naturally as unions of indistinguishable states, we implement a *disjoint-set forest* data structure Galler and Fisher (1964) to optimize both memory footprint and computational complexity.

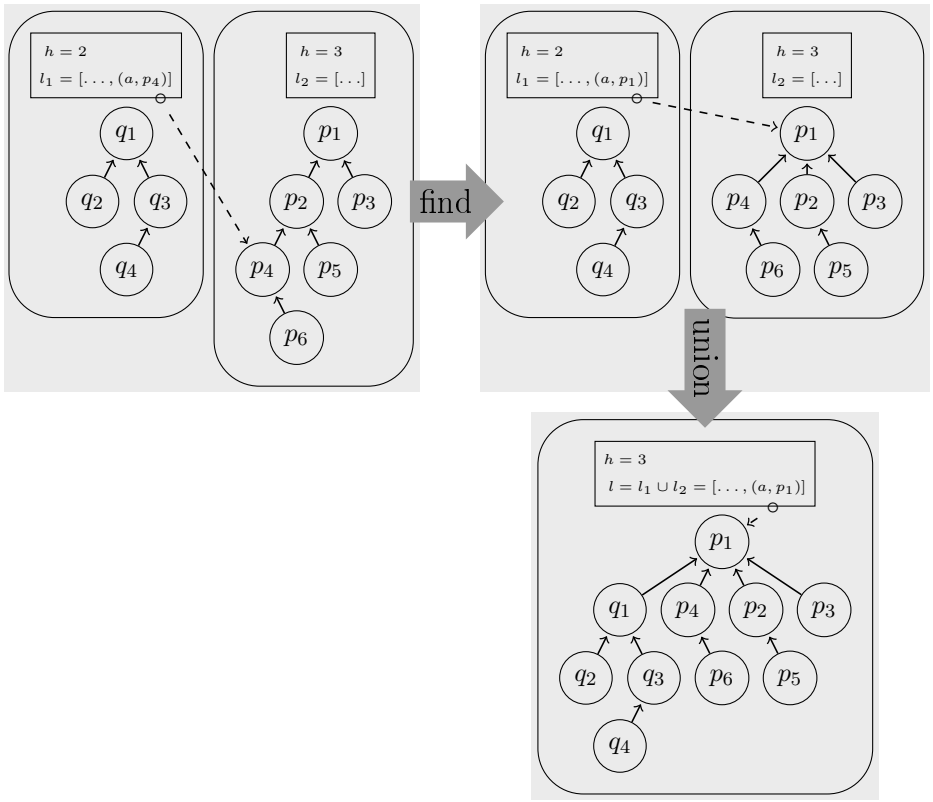

Figure 3.5: An example use of a merge-find structure while adding an epistemic class to the set of epistemic classes with a guaranteed winning strategy. The two stages of executing *union*(q_1, p_4) (finding the root of an epistemic class and joining two sets of classes) are shown. Note that an update of the arc labelled by a (between the pair of structures that contain q_1 and p_4) is also shown.

Structural Abstraction: Each epistemic equivalence class is represented as a tree within the disjoint-set forest (illustrated in Figure 3.5), where the *root node* serves as the canonical representative of its equivalence class and each node maintains:

- A parent pointer establishing the tree hierarchy
- An approximate rank (initially 1) for balancing operations
- A transition map encoding aggregated outgoing transitions at the class level

Core Operations: The optimization pipeline leverages three fundamental primitives:

`make_set(s)` Initializes singleton sets during model construction, assigning initial rank 1 to node s and establishing self-referential parent pointers.

`find(s)` Implements path compression: when locating s's representative, all nodes along the search path are directly linked to the root r, accelerating future queries. This operation returns r while implicitly updating stale transition endpoints through pointer redirection.

`union(`s_1`,`s_2`)` Merges two equivalence classes using union-by-rank:

1. The root with higher rank becomes the parent of the other
2. Upon rank equality, an arbitrary root is selected and its rank incremented
3. Outgoing transitions from merged classes are immediately redirected to the new representative
4. Incoming transitions employ lazy updates through subsequent `find` operations

Transition Management Strategy: The structure enables efficient transition bookkeeping:

- *Outgoing Transition Optimization:* When computing $\langle A \rangle^{\bullet} Q$, transitions from equivalence classes are precomputed and stored at the root, eliminating per-state edge validation.
- *Incoming Transition Compression:* For pre-image operations, incoming transitions are lazily updated via `find`'s path compression during their first use, ensuring amortized $O(\alpha(n))$ complexity for union and find operations, where α denotes the inverse Ackermann function Galler and Fisher (1964).

Complexity Analysis: This approach yields two key improvements:

1. Memory compression: Storage requirements reduce from $O(|S|)$ for explicit state sets to $O(|\mathcal{E}|)$, where $\mathcal{E}$ represents the number of epistemic equivalence classes.
2. Operation acceleration: Set union and membership checks improve from $O(n)$ to $O(\alpha(n))$, while transition updates benefit from batched pointer redirections during abstraction.

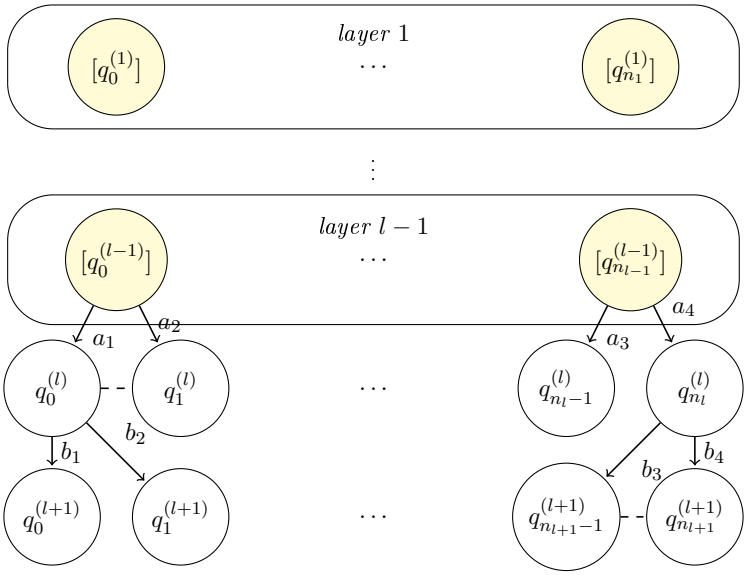

Figure 3.6: Intermediate shape of the state space for a bridge game during the model generation

Implementation Notes: The disjoint-set forest particularly benefits models with:

- Large observational symmetries (e.g., singleton coalitions with extensive indistinguishability neighborhoods)
- Dynamic coalition restructuring during verification
- High branching factors in transition systems where equivalence classes span multiple states

3.2.3 Dynamic Discarding of Irrelevant Submodels

While our proposed optimizations significantly enhance computational efficiency for strategic ability verification, their applicability to large-scale models remains constrained by memory bottlenecks inherent to explicit global state space generation. This limitation arises because the concurrent game structure (iCGS) often exhibits exponential state explosion relative to its parametric complexity, despite the subsequent epistemic abstraction reducing the effective state space.

Problem Statement: Current model generation algorithms operate in a two-phase paradigm:

1. *Global State Graph Construction:* The iCGS is fully instantiated *before* epistemic abstraction, requiring explicit storage of all states and transitions in memory.

2. *Redundant State Tracking:* To prevent duplicate state creation, the algorithm retains complete historical state information during stepwise generation from initial states, incurring $O(|S|)$ memory complexity for state set S.

This approach becomes infeasible for models where $|S|$ grows exponentially with domain parameters.

Solution: Layered State Generation with Epistemic Compression For models with stratified structure—such as the bridge endplay benchmark—we demonstrate that explicit state retention can be replaced by on-the-fly epistemic abstraction. These models exhibit:

- *Stratified Transition Dependency:* States in layer $l + 1$ depend only on states in layer l, forming a directed acyclic graph with layer-wise transition locality.

- *Epistemic Layer Isolation:* Observational indistinguishability relations remain intra-layer, as shown in Figure 3.6, enabling per-layer epistemic partitioning.

Our layered optimization modifies the generation algorithm to:

1. Process states incrementally by layer $l = 0, 1, ..., d$, where d is the model depth.

2. Apply epistemic abstraction to layer l immediately after its generation, collapsing states into observational equivalence classes.

3. Discard atomic state representations from layer $l - 1$ once layer l's transitions are resolved, retaining only their epistemic class descriptors for downstream computations.

This strategy reduces memory complexity from O(|S|) to O(d), where d represents the maximal aggregate width of two consecutive layers. For the bridge endplay benchmark, this corresponds to retaining only the epistemic signatures of the current and previous strata, rather than their full state expansions.

Generalization Potential: While the layered approach was developed for specific models, its applicability extends to any system with acyclic, locally bounded memory

requirements. Future work will investigate automated stratification detection and hybrid generation schemes that combine on-the-fly abstraction with traditional fixpoint algorithms.

3.3 Approximate Verification of Strategic Abilities under Imperfect Information Using Local Models

The formal verification of behavioral properties in multi-agent systems presents a significant computational challenge due to the inherent combinatorial explosion arising from interacting state spaces. Traditional model checking approaches require exhaustive analysis of the global state space, which grows exponentially with the number of agents, making verification intractable for complex systems. We introduce a novel verification paradigm that leverages agent-local models through contextual abstraction techniques. By constructing a bounded rationality approximation of the global system from the perspective of strategically selected agents, we maintain formal guarantees while substantially reducing computational complexity. Our approach preserves critical inter-agent dependencies through targeted observational equivalence criteria, enabling efficient verification of both safety and liveness properties with quantifiable accuracy bounds.

In this approach we now consider verification of *asynchronous* multi-agent systems given by modular representations, where agents operate without a global clock and may act independently, unlike in Sections 3.1 and 3.2 where synchronous systems were assumed.

3.3.1 Local Approximation Models of Ability

Definition 3.3.1 (Local Approximating Model) *The local approximating model for an agent i captures the essential behavior of the agent within the context of the entire system. This model is defined as a tuple $M_i = (L_i, Evtapp_i, R_i, PV_i, Tapp_i)$, where L_i represents the set of local states, $Evtapp_i$ the set of agent i events, R_i the repertoire function, PV_i the set of propositions, and $Tapp_i$ the transition relation. L_i, R_i, PV_i are defined as in standard AMAS per definition 2.1.7. The transition relation $Tapp_i$ is particularly crucial as it defines how the agent transitions between local states based*

on events. We define $Tapp_i$ in definition 3.3.2. Additionally, to account for the potential livelock for i, we introduce an auxliary event symbol τ, such that $Agents(\tau) = \emptyset$. Formally, $Evtapp_i = Evt_i \cup (\epsilon, \tau)$.

In this section, we formalize the new fixpoint approximation by defining the transition relation in the local approximating model and proving its properties. We also introduce the concepts of enabled events, standard outcomes, and reactive outcomes within this model. These definitions lay the groundwork for the execution semantics of the approximated model and the subsequent model checking procedures.

By leveraging the local approximating model, we aim to provide a scalable and efficient method for verifying multi-agent systems, ensuring that the verification process remains feasible even as the complexity of the system increases.

Definition 3.3.2 (Transition Relation in Local Approximating Model)
The transition relation $Tapp_i$ is defined as follows:

- $(l, \epsilon, l) \in Tapp_i$ *if there exist* $g \in St$ *such that* $g^i = l$ *and* $T(g, \epsilon) = g$.
- $(l, \tau, l) \in Tapp_i$ *if there exist* $g_1, g_2, \ldots, g_{n+1} \in St$ *and* $\alpha_1, \alpha_2, \ldots, \alpha_n \in Evt \setminus Evt_i$ *such that:*
  - $\forall j \in \{1, 2, \ldots, n\}, (g_j)^i = l$ *and* $T(g_j, \alpha_j) = g_{j+1}$,
  - $g_{n+1} = g_1$.

  In other words, there exists a sequence of global states in the global model that starts and ends in the global state containing the local state l, with all transitions in-between labeled by events that are not in the agent i's set of events.
- $(l, \alpha, l') \in Tapp_i$ *for* $\alpha \in Evt_i$ *if there exist* $g_1, g_2, \ldots, g_{n+1} \in St$ *and* $\alpha_1, \alpha_2, \ldots, \alpha_n \in Evt$ *such that:*
  - $\forall j \in \{1, 2, \ldots, n\}, (g_j)^i = l$ *and* $T(g_j, \alpha_j) = g_{j+1}$,
  - $\forall j \in \{1, 2, \ldots, n-1\}, i \notin Agent(\alpha_j)$,
  - $(g_{n+1})^i = l'$ *and* $\alpha_n = \alpha$.

  In other words, there exists a finite path fragment in the global model that runs entirely within the local state l (except possibly the last transition) and ends in a global state containing the local state l', with the last transition labeled by α and no other event executed by i.

The transition relation $Tapp_i$ captures the essential behavior of agent i within the global model. It defines how the agent transitions between local states based on events, ensuring that the local approximating model accurately represents the agent's interactions with the system. Consistent with the standard AMAS S, the transition relation is characterized as a partial transition function, defined for events within the agent's repertoire of choices.

Lemma 3.3.3 (Determinism in transition relation)
$\forall_{l\in L_i,\alpha\in Evtapp_i}(\exists_{l',l''\in L_i} : (l,\alpha,l') \in Tapp_i \wedge (l,\alpha,l'') \in Tapp_i) \Rightarrow l' = l''$

In order to talk about agent's strategies, we need to define the enabled events in the local approximating model. An event is enabled at a local state l if there exists a transition from l to another local state l' labeled with that event. This concept is essential for defining the standard and reactive outcomes of a strategy in the local approximating model.

Definition 3.3.4 (Enabled events in Local Approximating Model)
Event $\alpha \in Evtapp_i$ is enabled *at state $l \in L_i$ by choice $\beta \in R_i(l)$ in the local approximating model M_i if $l \xrightarrow{\alpha}_{M_i} l'$ for some $l' \in L_i$, i.e., $Tapp_i(l,\alpha) = l'$. The set of such events is denoted by $enabled_{M_i}(l, \sigma_i(l))$.*

The standard outcome captures the possible sequences of events that can be executed by the agent according to its strategy, while the reactive outcome further refines this by considering the enabled events at each local state.

Definition 3.3.5 (Standard outcome in Local Approximating Model)
Let $i \in$ A. The standard outcome *of strategy $\sigma_i \in \Sigma_i^{ir}$ in state $l \in L_i$ of the local approximating model M_i is the set $out^{\mathrm{Std}}_{M_i}(l,\sigma_i) \subseteq \Pi_{M_i}(l)$ such that $\pi = l_0\alpha_0 l_1\alpha_1 \cdots \in out^{\mathrm{Std}}_{M_i}(l,\sigma_i)$ iff $_0 = l$, and for each $m \geq 0$ we have that $\alpha_m \in enabled_{M_i}(l_m, \sigma_i(l_m))$.*

Definition 3.3.6 (Reactive outcome in Local Approximating Model)
Let $i \in$ A. The reactive outcome *of strategy $\sigma_i \in \Sigma_i^{ir}$ in state $l \in L_i$ of the local approximating model M_i is the set $out^{\mathrm{React}}_{M_i}(l,\sigma_i) \subseteq out^{\mathrm{Std}}_{M_i}(l,\sigma_i)$ such that $\pi = l_0\alpha_0 l_1\alpha_1 \cdots \in out^{\mathrm{React}}_{M_i}(l,\sigma_i)$ iff $\alpha_m = l\epsilon$ for each $m \geq 0$ implies $enabled_{M_i}(l_m, \sigma_i(l_m)) = \{\epsilon\}$.*

3.3.2 Fixpoint Approximation on Local Models

Now we can finally define the model checking procedure for the local approximating model. The model checking of a formula φ in the local approximating model M_i is performed by checking whether the formula holds for all sequences in the outcome of the agent's strategy in the initial state of the model.

Definition 3.3.7 (Model Checking Local Approximating Model)
Let $x \in \{\text{Std}, \text{React}\}$ and $i \in \text{A}$. $M_i, l \models^x \langle\langle i \rangle\rangle \varphi$ iff there is a strategy $\sigma_i \in \Sigma_i^{ir}$ such that for all $\pi \in out^x_{M_i}(l, \sigma_i)$ we have $M_i, \pi \models^x \varphi$.

To utilize the local approximating model for verifying the formula φ in the original global model, we must impose certain restrictions on the set of possible formulas. Specifically, all propositions that appear in φ must belong to the set PV_i of agent i. This requirement arises because the approximated model retains only the local states of the agent from the original model, omitting information about other agents. Consequently, we focus exclusively on singleton coalitions in our formulas. Additionally, similar to standard model checking for AMAS S, we restrict our attention to formulas that exclude next step operators $\mathbf{X}$ and nested strategic modalities. These constraints ensure the feasibility and accuracy of verification within the local approximating model framework.

Under the above constraints, we use the following translations of simple formulas of $\mathbf{ATL}_{\text{ir}}$, that provide the lower approximation.

Definition 3.3.8 (Lower Approximation for sATL $_{ir}$)

1. $tr_L(\langle\langle i \rangle\rangle \mathbf{F} \varphi) = \mu Z.(\varphi \vee \langle i \rangle Z)$;
2. $tr_L(\langle\langle i \rangle\rangle \mathbf{G} \varphi) = \nu Z.(\varphi \wedge \langle i \rangle Z$;
3. $tr_L(\langle\langle i \rangle\rangle \psi \, \mathbf{U} \, \varphi) = \mu Z.(\varphi \vee (\psi \wedge \langle i \rangle Z))$.

Next, we prove that, if $tr_L(\varphi)$ is satisfied in the corresponding local model, then φ must indeed hold in the global model of the system.

3.3.3 Correctness of the Approximation

Here, we present the main theoretical results of the paper, showing that procedure proposed in Section 3.3.2 provides a lower approximation of the verification problem for **ATL**$_{\text{ir}}$. To this end, we introduce the additional concept of a *partial ir-strategy*, which is simply a partial function $\sigma_i \colon L_i \rightharpoonup 2^{Evt_i}$ with the standard constraints. The outcome of such σ_i is defined analogously (note that it can contain infinite as well as finite paths!). Additionally, for $l \in L_i$ and $L \subseteq L_i$, we will write "$g \in l$" instead of "$g^i = l$," and "$g \in L$" instead of "$g \in l$ for some $l \in L$."

Theorem 3.3.9 (Approximation of reachability) *Let $x \in \{\text{Std}, \text{React}\}$, $i \in \text{A}$, and $l \in St_{M_i}$. If $M_i, l \models^x \mu Z.(\varphi \vee \langle i \rangle Z)$ then, for every $g \in l$, we have $M, g \models^x \langle\langle i \rangle\rangle \mathbf{F}\varphi$.*

We prove Theorem 3.3.9 as a direct consequence of the following lemma.

Lemma 3.3.10 *Let $Sat^x_n(M_i, \mu Z.(\varphi \vee \langle i \rangle Z))$ be the set of states in M_i, computed by the standard least fixpoint algorithm for $\mu Z.(\varphi \vee \langle i \rangle Z)$ with at most n fixpoint iterations (equivalently: with at most n executions of the preimage operation for $\langle i \rangle$).*

We claim that, for every $n \in \mathbb{N}$, if $L = Sat^x_n(M_i, \mu Z.(\varphi \vee \langle i \rangle Z))$ then, for every $g \in L$, we have $M, g \models^x \langle\langle i \rangle\rangle \mathbf{F}\varphi$.

Proof. Proof by induction on n.

Base Case, $n = 0$: If $n = 0$, then $L = Sat^x_0(M_i, \mu Z.(\varphi \vee \langle i \rangle Z))$ contains only the states where φ is satisfied. Therefore, for every $g \in St_M$ such that $g^i = l \in L$, we have $M, g \models^x \varphi$. Hence, $M, g \models^x \langle\langle i \rangle\rangle \mathbf{F}\varphi$.

Inductive Step: Assume that the lemma holds for some $n \geq 0$. We need to show that it also holds for $n + 1$.

Let $L_{n+1} = Sat^x_{n+1}(M_i, \mu Z.(\varphi \vee \langle i \rangle Z))$. By the definition of the least fixpoint algorithm, L_{n+1} is the set of states where either φ is satisfied or there exists a choice in the repertoire of i such that every resulting transition leads to a state in L_n. Formally, $L_{n+1} = \{l \mid M_i, l \models \varphi \vee \exists_{\beta \in R_i(l)} \forall_{\alpha \in enabled_M(l,\beta)} Tapp_i(l, \alpha) \in L_n\}$.

Consider any $g \in St_M$ such that $g^i = l \in L$. We have two cases to consider:

1. $M_i, l \models \varphi$: In this case, $M, g \models^x \varphi$, and hence $M, g \models^x \langle\langle i \rangle\rangle \mathbf{F}\varphi$.

2. $\exists_{\beta \in R_i(l)} \forall_{\alpha \in enabled_M(l,\beta)} Tapp_i(l, \alpha) \in L_n$: By the inductive hypothesis, for every $g' \in St_M$ such that $g'^i = l'$, we have $M, g' \models^x \langle\langle i \rangle\rangle \mathbf{F}\varphi$.

We proceed as follows:

(i) By the induction lemma, we obtain a partial strategy σ_a defined on L_n, such that for every global state $g \in L_n$ and path $\lambda \in out^x(g, \sigma_a)$, the path λ is infinite and there exists an index $i \in \mathbb{N}$ such that $M, \lambda[i] \models \varphi$.

(ii) From the $(n+1)$-th iteration, we derive a partial strategy σ'_a defined on $L_{n+1} \setminus L_n$, such that for every global state $g \in L_{n+1} \setminus L_n$ and path $\lambda \in out^x(g, \sigma'_a)$, the path λ is finite and $last(\lambda) \in L_n$.

(iii) Since σ_a and σ'_a are defined on disjoint subsets of locations, they can be combined into a new strategy $\sigma''_a = \sigma_a \cup \sigma'_a$. Consequently, every path $\lambda'' \in out^x(g, \sigma''_a)$ is infinite and consists of the concatenation of a finite prefix $\lambda' \in out^x(g, \sigma'_a)$ with an infinite path $\lambda \in out^x(\hat{g}, \sigma_a)$ for some $\hat{g} \in L_n$. Therefore, there must exist some index $j \in \mathbb{N}$ such that $M, \lambda[j] \models \varphi$, which completes the proof that $M, g \models^x \langle\langle i \rangle\rangle \mathbf{F}\varphi$.

Therefore, by induction, the claim holds for all $n \in \mathbb{N}$. □

Theorem 3.3.11 (Approximation of safety) *Let $x \in \{\text{Std}, \text{React}\}$, $i \in \text{A}$, and $l \in St_{M_i}$. If $M_i, l \models^x \nu Z.(\varphi \wedge \langle i \rangle Z)$ then, for every $g \in l$, we have $M, g \models^x \langle\langle i \rangle\rangle \mathbf{G}\varphi$.*

Theorem 3.3.11 is a direct consequence of Lemma 3.3.12, that uses the *"bounded always"* operator $\mathbf{G}^n$ with semantics given by:

$M, q \models^x \langle\langle A \rangle\rangle \mathbf{G}^n \varphi$ iff there is a strategy $s_A \in \Sigma^{ir}_A$ such that, for each path $\lambda \in out^x_M(q, s_A)$, we have $M, \lambda[i] \models^x \varphi$ for all $0 \leq i \geq n$.

Lemma 3.3.12 *Let $Sat^x_n(M_i, \nu Z.(\varphi \wedge \langle i \rangle Z))$ be the set of states in M_i, computed by the standard greatest fixpoint algorithm for $\nu Z.(\varphi \wedge \langle i \rangle Z)$ with at most n fixpoint iterations.*

We claim that, for every $n \in \mathbb{N}$, if $L = Sat^x_n(M_i, \nu Z.(\varphi \wedge \langle i \rangle Z))$ then, for every $g \in L$, we have $M, g \models^x \langle\langle i \rangle\rangle \mathbf{G}^n \varphi$.

Proof. We prove this lemma by induction on n.

Base Case, $n = 0$: If $n = 0$, then $L = Sat^x_0(M_i, \nu Z.(\varphi \wedge \langle i \rangle Z))$ contains only the states where φ is satisfied. Therefore, for every $g \in St_M$ such that $g^i = l \in L$, we have $M, g \models^x \varphi$. Hence, $M, g \models^x \langle\langle i \rangle\rangle \mathbf{G}^0 \varphi$.

Inductive Step: Assume the lemma holds for n, i.e., for every $g \in L_n$, $M, g \models_x \langle\langle i \rangle\rangle G_n \varphi$. We need to show it holds for $n + 1$.

Consider the set $L_{n+1} = Sat^x_{n+1}(M_i, \nu Z.(\varphi \wedge \langle i \rangle Z))$. By construction:

$$L_{n+1} \subseteq L_n$$

and

$$L_{n+1} = \{l \in L_i \mid \exists \sigma_i \in \Sigma^i_{ir} \text{ s.t. } \forall \pi \in \text{out}_x(l, \sigma_i), \pi[0] \in L_n\}.$$

For any $g \in L_{n+1}$, there exists a strategy σ'_i for agent i such that:

1. $\sigma'_i(l)$ selects an event $\alpha \in \text{enabled}_{M_i}(l)$ transitioning to $l' \in L_n$.
2. φ holds at $\pi[0]$ for all paths $\pi \in \text{out}_x(l, \sigma'_i)$.

We define the combined strategy σ''_i recursively:

$$\sigma''_i(l) = \begin{cases} \sigma'_i(l) & \text{if } l \in L_{n+1} \setminus L_n, \\ \sigma_i(l) & \text{if } l \in L_n, \end{cases}$$

where σ_i is the strategy guaranteed by the inductive hypothesis for L_n.

Correctness of σ''_i:

- **First Step:** From $l \in L_{n+1}$, σ''_i transitions to $l' \in L_n$, ensuring φ holds at the next state.
- **Inductive Step:** From $l' \in L_n$, σ_i ensures φ holds for n subsequent steps.

Thus, for any $g \in L_{n+1}$:

$$M, g \models_x \langle\langle i \rangle\rangle G_{n+1} \varphi \quad \text{via } \sigma''_i.$$

Therefore, by induction, the claim holds for all $n \in \mathbb{N}$. □

The characterization for "until" formulas only slightly extends that of reachability (Theorem 3.3.9). We only mention the theorem and the main lemma here, as the rest of the proof is completely analogous.

Theorem 3.3.13 (Approximation of until) *Let $x \in \{\text{Std}, \text{React}\}$, $i \in \text{A}$, and $l \in St_{M_i}$. If $M_i, l \models^x \mu Z.(\varphi \vee (\psi \wedge \langle i \rangle Z))$ then, for every $g \in l$, we have $M, g \models^x \langle\langle i \rangle\rangle \psi \, \mathbf{U} \, \varphi$.*

Lemma 3.3.14 *Let $Sat^x_n(M_i, \mu Z.(\varphi \vee (\psi \wedge \langle i \rangle Z)))$ be the set of states in M_i, computed by the standard least fixpoint algorithm for $\mu Z.(\varphi \vee (\psi \wedge \langle i \rangle Z))$ with at most n fixpoint iterations (equivalently: with at most n executions of the preimage operation for $\langle i \rangle$).*

We claim that, for every $n \in \mathbb{N}$, if $L = Sat^x_n(M_i, \mu Z.(\varphi \vee (\psi \wedge \langle i \rangle Z))$ then, for every $g \in L$, we have $M, g \models^x \langle\langle i \rangle\rangle \psi \, \mathbf{U} \, \varphi$.

Proof. Proof by induction on n.

Base Case, $n = 0$: If $n = 0$, then $L = Sat^x_0(M_i, \mu Z.(\varphi \vee (\psi \wedge \langle i \rangle Z)))$ contains only the states where φ is satisfied. Therefore, for every $g \in St_M$ such that $g^i = l \in L$, we have $M, g \models^x \varphi$. Hence, $M, g \models^x \langle\langle i \rangle\rangle \psi \, \mathbf{U} \, \varphi$.

Inductive Step: Assume that the lemma holds for some $n \geq 0$. We need to show that it also holds for $n + 1$.

Let $L_{n+1} = Sat^x_{n+1}(M_i, \mu Z.(\varphi \vee (\psi \wedge \langle i \rangle Z)))$. By the definition of the least fixpoint algorithm, L_{n+1} is the set of states where either φ is satisfied or ψ is satisfied and there exists a transition to a state in L_n. Formally, $L_{n+1} = \{l \mid M_i, l \models \varphi \vee (M_i, l \models \psi \wedge \exists_{\beta \in R_i(l)} \forall_{\alpha \in enabled_M(l,\beta)} Tapp_i(l, \alpha) \in L_n)\}$.

Consider any $g \in St_M$ such that $g^i = l \in L$. We have two cases to consider:

1. $M_i, l \models \varphi$: In this case, $M, g \models^x \varphi$, and hence $M, g \models^x \langle\langle i \rangle\rangle \psi \, \mathbf{U} \, \varphi$.

2. $M_i, l \models \psi \wedge \exists_{\beta \in R_i(l)} \forall_{\alpha \in enabled_M(l,\beta)} Tapp_i(l, \alpha) \in L_n$: By the inductive hypothesis, for every $g' \in St_M$ such that $g'^i = l'$, we have $M, g' \models^x \langle\langle i \rangle\rangle \psi \, \mathbf{U} \, \varphi$.

We proceed as follows:

(i) By the induction lemma, we obtain a partial strategy σ_a defined on L_n, such that for every global state $g \in L_n$ and path $\lambda \in out^x(g, \sigma_a)$, the path λ is infinite and there exists an index $i \in \mathbb{N}$ such that $M, \lambda[i] \models \varphi$ and $\forall_{0 \leq j < i} M, \lambda[j] \models \psi$.

(ii) From the $(n+1)$-th iteration, we derive a partial strategy σ'_a defined on $L_{n+1} \setminus L_n$, such that for every global state $g \in L_{n+1} \setminus L_n$ and path $\lambda \in out^x(g, \sigma'_a)$, the path λ is finite and $last(\lambda) \in L_n$.

(iii) Since σ_a and σ'_a are defined on disjoint subsets of locations, they can be combined into a new strategy $\sigma''_a = \sigma_a \cup \sigma'_a$. Consequently, every path $\lambda'' \in out^x(g, \sigma''_a)$ is infinite and consists of the concatenation of a finite prefix $\lambda' \in out^x(g, \sigma'_a)$ with an infinite path $\lambda \in out^x(\hat{g}, \sigma_a)$ for some $\hat{g} \in L_n$. Therefore, there must exist some index $j \in \mathbb{N}$ such that $M, \lambda[j] \models \varphi$ and $\forall_{0 \leq k < j} M, \lambda[k] \models \psi$, which completes the proof that $M, g \models^x \langle\langle i \rangle\rangle \psi \, \mathbf{U} \, \varphi$.

Therefore, by induction, the claim holds for all $n \in \mathbb{N}$. □

3.4 Challenges and Lessons Learnt

The development and implementation of fixpoint-based approximations for ATL$_{ir}$ model checking revealed several critical challenges and corresponding insights. These lessons inform both theoretical advancements and practical applications of strategic ability verification in multi-agent systems.

The experimental evaluation of the proposed approximation techniques across diverse benchmark models is presented in Chapter 9.

Challenges

1. **Semantic Mismatch in Multi-Agent Coalitions:** The failure of fixpoint equivalences in ATL$_{ir}$ compared to perfect information settings (Proposition 3.1.1) necessitated entirely new translation strategies for coalitions with imperfect information. Naive translations like trL1 and trL2 (Section 3.1.1) proved insufficient for multi-agent scenarios, as they neither overapproximated nor underapproximated abilities (Figures 3.1 and 3.2).

2. **Tightness vs. Computational Tractability:** Achieving tight lower bounds required balancing strict common knowledge constraints (trL3) with the need to allow flexible step counts within epistemic neighborhoods (Remark 3.1.6). Proposition 3.1.9 demonstrated that non-tight approximations arise when agents lose information during strategy execution, creating a disconnect between local reasoning and global outcomes.

3. **Exponential State Space Complexity:** Explicit global state generation for iCGS led to infeasible memory usage for large models. Layered state structures (Figure 3.6) required novel compression techniques to avoid storing redundant historical states.

4. **Dynamic Epistemic Abstraction:** Managing transitions between epistemic equivalence classes while preserving observational history demanded careful integration of disjoint-set data structures (Figure 3.5), where path compression and lazy updates introduced implementation complexity.

Lessons Learnt

1. **Common Knowledge as a Semantic Anchor:** Replacing "everybody knows" neighborhoods with common knowledge neighborhoods (Section 3.1.4.1) proved essential for universal lower bounds. This highlighted the importance of epistemic depth in strategic reasoning.

2. **Lockstep Models Simplify Verification:** For agents who observe all actions (Proposition 3.1.8), standard and steadfast operators coincide. This suggests that domain-specific knowledge about observability can reduce computational overhead without sacrificing accuracy.

3. **Recomputable Strategies Enable Tighter Bounds:** The necessity of recomputable strategies (Proposition 3.1.9) for tight lower bounds revealed that agents must retain sufficient information during execution to reassess their plans. This aligns with practical limitations in real-world systems where memory constraints affect strategic adaptability.

4. **Structural Exploitation Reduces Complexity:** Stratified dependencies (e.g., bridge endplay models) allowed layer-wise processing and dynamic discarding of obsolete states (Section 3.2.3), reducing memory complexity from exponential to linear in model depth. Disjoint-set optimizations (Section 3.2.2) demonstrated that union-find heuristics (path compression, union-by-rank) can achieve near-constant time complexity for set operations, even in large-scale imperfect information settings.

5. **Approximation Semantics Trade Precision for Scalability:** The proposed trL and trU translations (Section 3.1.6) enable efficient verification at the cost of exactness, but their correctness guarantees (Theorem 3.1.11) ensure no false negatives (for trL) or false positives (for trU). For singleton coalitions, the lower bound trL3 achieves optimal performance, as shown in Proposition 3.1.7, where $\langle A \rangle^\bullet$ outperforms $\langle A \rangle$ in tightness.

6. **Perfect Recall Can Be Emulated via State Encoding:** Many benchmark models (e.g., bridge endplay) achieve perfect recall through explicit observational history in state representations. This collapses the distinction between memoryless and perfect recall semantics, making trL3 applicable without modification (Section 3.1.7).

7. **Local Models Reduce State Explosion:** The introduction of local approximation models (Section 3.3.1) allows for modular verification that avoids the state explosion associated with global models. This approach leverages asynchronous system representations to improve scalability while maintaining soundness of the approximations.

These challenges and lessons underscore the interplay between epistemic structure, strategic flexibility, and computational efficiency. Future work will focus on automated stratification detection and hybrid verification frameworks combining fixpoint approximations with exact methods for critical subsystems.

4

Forward-Chaining Strategy Synthesis

This chapter outlines the development of a forward-chaining strategy synthesis algorithm that is anchored in the classical Depth First Search (DFS) methodology. The conventional recursive DFS approach required significant adaptations to accommodate the unique challenges presented by epistemic classes and the nondeterministic outcomes characteristic of coalition actions. Specifically, the inherent nondeterminism, resulting from multiple potential transitions, introduces a scenario where decisions in one branch may necessitate adjustments in parallel branches. This is particularly pertinent when a locally winning strategy, viable within a singular branch, demands reconsideration due to its implications on choices within nodes of the same epistemic class, thereby affecting transition outcomes across the model.

To navigate these complexities, the algorithm is designed to facilitate strategic backtracking, enabling the revision of locally winning strategies in the event that a comprehensive winning strategy remains elusive in another branch. Given the vast strategy space associated with even modestly sized models—those comprising hundreds of states—the exhaustive search becomes impractical. Consequently, this necessitates the employment of the DFS algorithm as a foundational framework upon which more sophisticated techniques are implemented to efficiently manage and navigate the strategy space.

The content of this chapter is based on the following papers: Jamroga et al. (2022b); Kurpiewski et al. (2019b).

4.1 Strategy Synthesis Based on the DFS-Algorithm

Within our formal framework, we focus on the property of *enforceability*. A proposition $p \in \mathit{Props}$ is enforceable from state $q \in \mathit{St}$ by coalition $A \subseteq \mathbb{A}\mathrm{gt}$ via strategy $\sigma_A \in \Sigma_A$ if every infinite path $\lambda \in \mathit{out}(q, \sigma_A)$ satisfies $\lambda_i \in V(p)$ for some $i \in \mathbb{N}$. This relation is denoted $q \models \langle \sigma_A \rangle Fp$, while $q \models \langle\langle A \rangle\rangle Fp$ signifies existence of such a strategy.

Our approach diverges from traditional methods by constructing winning strategies through composition of conflictless partial strategies. We formalize this process as follows:

Definition 4.1.1 *For conflictless partial strategies $\sigma_A, \sigma'_A \in \Sigma_A$ (where $\mathit{dom}(\sigma_A) \cap \mathit{dom}(\sigma'_A) = \emptyset$ and $\sigma_A \cup \sigma'_A$ forms a partial strategy), we define:*

- Exit set*:* $\mathcal{O}(\sigma_A) = \{q' \in \mathit{St} \setminus \mathit{dom}(\sigma_A) \mid \exists q \in \mathit{dom}(\sigma_A)\, \exists \beta \in d_{\mathbb{A}\mathrm{gt}\setminus A}(q) : o(q, (\sigma_A(q), \beta)) = q'\}$*, representing states reachable from σ_A but outside its domain.*

- Shared control set*:* $\mathcal{SC}(\sigma_A, \sigma'_A) = \{q \in \mathit{bord}(\sigma_A) \cup \mathit{bord}(\sigma'_A) \mid \forall a \in A\, \exists q' \in \mathit{dom}(\sigma_A) \cup \mathit{dom}(\sigma'_A) : q \sim_a q'\}$*, capturing states where both strategies maintain coalition control.*

- Shared-control strategy*:*
  $\Delta(\sigma_A, \sigma'_A) = \{(q, \alpha) \in d_A(q) \mid q \in \mathcal{SC}(\sigma_A, \sigma'_A)$ *and α is induced by σ_A or σ'_A}*, integrating control mechanisms from both strategies.*

- Composition operators*:*

$$\sigma_A \uplus \sigma'_A = \sigma_A \cup \sigma'_A \cup \Delta(\sigma_A, \sigma'_A)$$
$$\Lambda(\sigma_A, \sigma'_A) = \sigma_A \cup \Delta(\sigma_A, \sigma'_A)$$

  extending strategies with shared-control components.

- Input set*:* $\mathcal{I}(\sigma_A, \sigma'_A) = (\mathcal{O}(\sigma'_A) \cap \mathit{dom}(\Lambda(\sigma_A, \sigma'_A))) \cup q^e_{\mathit{init}}$*, where $q^e_{\mathit{init}} = \{q_{\mathit{init}}\}$ if $q_{\mathit{init}} \in \mathit{dom}(\sigma_A)$ and $\emptyset$ otherwise. This characterizes states where σ'_A transitions into σ_A's operational domain.*

Notably, enforceability requires strategies that both avoid infinite loops and guarantee proposition satisfaction. This yields:

Proposition 4.1.2 (Kurpiewski et al. (2019b)) *For any coalition $A \subseteq \mathbb{A}\mathrm{gt}$, $q \models \langle\langle A \rangle\rangle Fp$ holds if and only if there exists a p-free partial strategy $\sigma_A \in \Sigma_A$ such that:*

1. *$q \in dom(\sigma_A)$,*
2. *σ_A is q-loopless (contains no cycles in its execution paths),*
3. *$\mathcal{O}(\sigma_A) \subseteq V(p)$ (all exit states satisfy p).*

4.1.1 Substituting Strategies

In the remainder of this section, we formalize the following conceptual framework. Consider two conflictless partial strategies, σ_A^C (complementary) and σ_A (primary), governing the system's execution. The combined domain of these strategies is explicitly defined, while the objectives to be enforced reside strictly outside their operational scope. For simplicity, assume empty boundary conditions for both strategies.

We formalize the *inputs* of σ_A^C relative to σ_A as the set of states reachable under σ_A^C's control that transition into σ_A's domain. Correspondingly, the *outputs* of σ_A are characterized as the terminal states encountered immediately upon exiting its domain after trajectories initiated from a given input. This mapping constitutes the *input/output characteristic* of σ_A with respect to σ_A^C, denoted $\mathcal{I}/\mathcal{O}_{\sigma_A|\sigma_A^C}$.

Crucially, the system's objective is jointly enforced by σ_A^C and σ_A if all execution paths originating from their domain union satisfy the goal specification. Under this foundational assumption, we establish that a replacement partial strategy σ'_A preserving the input set of σ_A while providing enhanced output regulation can substitute σ_A without compromising enforcement guarantees relative to σ_A^C.

Definition 4.1.3 (Input/Output Characteristic) *Let $\sigma_A, \sigma_A^C \in \Sigma_A$ be conflictless. The* input/output characteristic of σ_A w.r.t. σ_A^C *is defined as a function $\mathcal{IO}(\sigma_A, \sigma_A^C) \colon \mathcal{I}(\sigma_A, \sigma_A^C) \to 2^{\mathcal{O}(\Lambda(\sigma_A, \sigma_A^C))}$ such that for each $q \in \mathcal{I}(\sigma_A, \sigma_A^C)$ we have $\mathcal{IO}(\sigma_A, \sigma_A^C)(q) = \mathcal{O}(\Lambda(\sigma_A, \sigma_A^C)) \cap (\mathbf{str}_{out(q, \Lambda(\sigma_A, \sigma_A^C))})$.*

We formalize this framework through the following construction: assume disjoint control between strategies σ_A and σ_A^C, formally $\Lambda(\sigma_A, \sigma_A^C) = \sigma_A$. Define $\mathcal{O}(\sigma_A)$ as the set of terminal states reached under σ_A's execution that lie outside its domain. The support $(\mathbf{str}_{out(q,\sigma_A)})$ captures all states along trajectories initiated at q under σ_A's governance. Crucially, the input/output characteristic $\mathcal{IO}(\sigma_A, \sigma_A^C)(q)$ specifies the

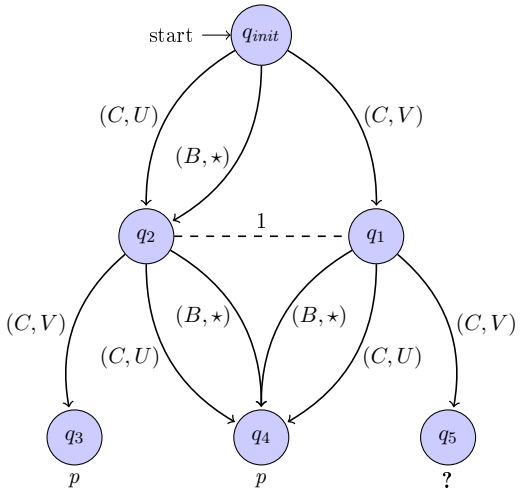

Figure 4.1: Example iCGS

minimal set of exit states encountered immediately after departing from σ_A's domain when executing its prescribed policy from input state q.

The following example illustrates the definition.

Example 4.1.4 *Fig. 4.1 illustrates an iCGS comprising six states and two agents. Solid edges represent temporal transitions, annotated with agent-specific action pairs. The $\star$ symbol denotes agent-agnostic choices. Dotted equivalence classes indicate observational indistinguishability for Agent 1.*

In state q_{init}, Agent 1 may either transition to q_2 via B or relinquish control to the co-agent for q_1/q_2 selection. Notably, q_1 and q_2 form an observational equivalence class for Agent 1. Within this class, Agent 1 can prescribe uniform execution of C or B across all states in its strategy framework. When enforcing C, the co-agent determines transitions to $\{q_3, q_4\}$ (from q_2) or $\{q_4, q_5\}$ (from q_1). Conversely, B enforcement deterministically routes to q_4 irrespective of co-agent decisions. Proposition p holds in q_3 and q_4; the labeling of q_5 remains open for subsequent specification.

Consider the singleton coalition $A = \{1\}$ with three partial strategies: σ_A^C, σ_A, and σ'_A satisfying $dom(\sigma_A^C) = \{q_{init}\}$ and $dom(\sigma_A) = dom(\sigma'_A) = \{q_2\}$. Define $\sigma_A^C(q_{init}) = C$, $\sigma_A(q_2) = B$, and $\sigma'_A(q_2) = C$.

This yields $\mathcal{O}(\sigma_A^C) = \{q_1, q_2\}$ with control casts: $\Lambda(\sigma_A, \sigma_A^C)(q_i) = B$ and $\Lambda(\sigma'_A, \sigma_A^C)(q_i) = C$ for $i \in \{1, 2\}$. Consequently, both strategies share identical input sets: $\mathcal{I}(\sigma'_A, \sigma_A^C) = \mathcal{I}(\sigma_A, \sigma_A^C) = \{q_1, q_2\}$. Their input/output characteristics diverge: $\mathcal{IO}(\sigma_A, \sigma_A^C)(q_1) = \mathcal{IO}(\sigma_A, \sigma_A^C)(q_2) = \{q_4\}$, while $\mathcal{IO}(\sigma'_A, \sigma_A^C)(q_1) = \{q_4, q_5\}$ and $\mathcal{IO}(\sigma'_A, \sigma_A^C)(q_2) = \{q_3, q_4\}$.

4.2 How to Exploit Strategic Dominance

In game theory, the iterative removal of dominated strategies serves as a fundamental technique for reducing the strategy profile space relevant to solution concepts such as Nash equilibrium. Specifically, when a strategy yields strictly inferior payoffs compared to an alternative strategy, the dominated strategy can be safely excluded from further analysis. We begin by formalizing the notion of strategy comparison within our framework.

4.2.1 Comparing Partial Strategies

We now provide a method for comparing partial strategies in the presence of a context, based on their input/output characteristics.

Definition 4.2.1 (Partial Strategy Comparison Framework)
For strategies $\sigma_A, \sigma'_A, \sigma_A^C \in \Sigma_A$ where σ_A and σ'_A are conflictless with respect to σ_A^C, we define σ_A^C as the context strategy. *The refinement relation $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$ holds iff:*

1. *$dom(\mathcal{IO}(\sigma_A, \sigma_A^C)) = dom(\mathcal{IO}(\sigma'_A, \sigma_A^C))$, meaning σ_A and σ'_A maintain identical input domains relative to σ_A^C; and*

2. *$\forall q \in dom(\mathcal{IO}(\sigma_A, \sigma_A^C))$, $\mathcal{IO}(\sigma_A, \sigma_A^C)(q) \subseteq \mathcal{IO}(\sigma'_A, \sigma_A^C)(q)$, indicating σ_A ensures output containment within the characteristic of σ'_A.*

When σ_A^C is fixed, $\preceq_{\sigma_A^C}$ establishes a preorder over conflictless partial strategies. While antisymmetry generally fails, we note that $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$ and $\sigma_A \preceq_{\sigma_A^C} \sigma'_A$ collectively imply $\mathcal{IO}(\sigma_A, \sigma_A^C) = \mathcal{IO}(\sigma'_A, \sigma_A^C)$, equating their input/output characteristics.

We now remove the need to specify the context.

Definition 4.2.2 (Context-Free Strategy Comparison) *Define the set of* strategic coverings *as:*

$$\Gamma_A = \left\{ \{\sigma_A, \sigma_A^C\} \subseteq \Sigma_A \,\middle|\, \begin{array}{l} \sigma_A, \sigma_A^C \text{ are conflictless, } \Lambda(\sigma_A, \sigma_A^C) \text{ is p-free,} \\ \mathcal{I}(\sigma_A, \sigma_A^C)\text{-loopless, and } q_{init} \in dom(\sigma_A) \cup dom(\sigma_A^C) \end{array} \right\}.$$

For strategic coverings $\gamma_A = \{\sigma_A, \sigma_A^C\}$ and $\gamma'_A = \{\sigma'_A, \sigma_A^C\}$ in Γ_A, we define the relation $\gamma_A \preceq_0 \gamma'_A$ iff $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$. The extended relation $\preceq$ is then established as the reflexive and transitive closure of $\preceq_0$.

This comparison mechanism induces a preorder over the space of strategic coverings. While we have explicitly defined $\preceq$ for pairs $\{\sigma_A, \sigma_A^C\}$, its generalization to arbitrary strategy tuples follows naturally through analogous constraints and is therefore omitted for notational brevity.

So far we have introduced notions that allow for comparing strategic coverings. We can now show that the relation of comparison indeed preserves enforceability. Intuitively, a stronger strategy (w.r.t. $\preceq$) can achieve at least what the weaker one can. For brevity, we write $\bigcup \gamma_A = \sigma_A^C \cup \sigma_A$ for each $\gamma_A = \{\sigma_A, \sigma_A^C\} \in \Gamma_A$.

Theorem 4.2.3 (On Substituting Strategies Kurpiewski et al. (2019b)) *Let $\gamma_A, \gamma'_A \in \Gamma_A$ be such that $\gamma'_A \preceq \gamma_A$. If $q_{init} \models \langle\bigcup \gamma'_A\rangle Fp$, then also $q_{init} \models \langle\bigcup \gamma_A\rangle Fp$.*

The proof can be found in Kurpiewski et al. (2019b).

Let us illustrate the theorem with an example.

Example 4.2.4 *Building upon Example 4.1.4 and Fig. 4.1, we introduce strategy ${\sigma'}_A^C \in \Sigma_A$ with $dom({\sigma'}_A^C) = \{q_{init}\}$ and ${\sigma'}_A^C(q_{init}) = B$. The refinement relations emerge as follows:*

- *$\{\sigma_A^C, \sigma'_A\} \preceq \{\sigma_A^C, \sigma_A\}$ holds via $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$ (established in Example 4.1.4);*
- *$\{\sigma_A^C, \sigma_A\} \preceq \{{\sigma'}_A^C, \sigma_A\}$ follows from $\sigma_A^C \preceq_{\sigma_A} {\sigma'}_A^C$;*
- *Transitive closure yields $\{\sigma_A^C, \sigma'_A\} \preceq \{{\sigma'}_A^C, \sigma_A\}$.*

The strategic configuration $\{\sigma_A^C, \sigma_A\}$ (always executing C) refines $\{{\sigma'}_A^C, \sigma_A\}$ (always executing B) under $\preceq$. Notably, B-dominated strategies guarantee p-enforcement from q_{init} even when q_5 remains unlabeled, demonstrating the robustness of context-free strategy dominance in achieving objective specifications.

4.2.2 Complexity

Definition 4.2.5 (Strict Contextual Dominance) *For strategies σ_1 and σ_2, we define $\sigma_1 \prec_{\sigma_A^C} \sigma_2$ if $\sigma_1 \preceq_{\sigma_A^C} \sigma_2$ holds while $\sigma_2 \preceq_{\sigma_A^C} \sigma_1$ does not.*

Definition 4.2.6 (Optimality) *A strategy σ_1 is* optimal *relative to context σ^C if no alternative strategy σ_2 exists satisfying $\sigma_1 \prec_{\sigma_A^C} \sigma_2$.*

Theorem 4.2.7 (Optimality Complexity Kurpiewski et al. (2019b)) *The decision problem of determining whether a partial strategy is optimal for a given context and iCGS is* **co-NP***-complete.*

The proof establishes polynomial-time verifiability of counterexamples and reduces from the complement of the 3-SAT problem Kurpiewski et al. (2019b).

4.2.3 On Independence of Components

This subsection establishes theoretical foundations for parallel strategy synthesis through strategic covering decomposition. We formalize conditions enabling distributed computation of maximal strategies:

Definition 4.2.8 (Strategic Covering Decomposition) *A strategic covering* $\gamma_A = \{\sigma_A^0, \sigma_A^1\}$ *admits parallel processing if two independent processes satisfy:*

1. *Each process* $i \in \{0,1\}$ *computes a maximal strategy* $\sigma_A^{m,i}$ *such that* $\sigma_A^i \preceq_{\sigma_A^C} \sigma_A^{m,i}$ *relative to context* $\sigma_A^{(i+1) \bmod 2}$
2. *The resulting covering* $\gamma_A^m = \{\sigma_A^{m,1}, \sigma_A^{m,2}\}$ *maintains dominance over the original* γ_A

In preorders, maximality requires that no strictly dominating element exists:

Definition 4.2.9 (Maximal Element) *Element* e *in preorder* $\sqsubset$ *is maximal iff* $\forall e' \in E : e \sqsubset e' \Rightarrow e' \sqsubset e$

Strategic independence emerges when agent knowledge doesn't overlap:

Definition 4.2.10 (Contextual Independence) *Strategy* $\sigma_A \in \Sigma_A$ *is* σ_A^C-independent *iff* $\nexists q \in dom(\Lambda(\sigma_A, \sigma_A^C)), q' \in dom(\sigma_A^C), a \in A : q \sim_a q'$

The following lemma reveals critical properties of context transformations:

Lemma 4.2.11 (Contextual Independence Preservation Kurpiewski et al. (2019b)) *Let* $\sigma_A, \sigma'_A \in \Sigma_A$ *be* σ_A^C*- and* σ_A^{Cm}*-independent strategies. If* $\mathcal{O}(\sigma_A^{Cm}) \subseteq \mathcal{O}(\sigma_A^C)$ *and* $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$*, then* $\sigma'_A \preceq_{\sigma_A^{Cm}} \sigma_A$ *holds.*

The proof demonstrates preorder invariance under context contraction Kurpiewski et al. (2019b). This result enables domain partitioning through common knowledge

equivalence classes: when $[dom(\sigma_A)]_{\sim_A^C}$ and $[dom(\sigma_A^C)]_{\sim_A^C}$ are disjoint, mutual independence naturally follows. This decomposition pattern mirrors bisimulation-based approximations in strategic logics Belardinelli et al. (2017a); Jamroga et al. (2017a).

The next theorem points to some sources of conflict between two strategies, leading to a natural observation that a strategy can be safely replaced with a stronger one, if the observed change is not visible outside of its domain.

Theorem 4.2.12 (Component-wise Comparison Kurpiewski et al. (2019b)) *Let* $\gamma_A, \gamma_A^m \in \Gamma_A$ *be s.t.* $\gamma_A = \{\sigma_A, \sigma_A^C\}$ *and* $\gamma_A^m = \{\sigma_A^m, \sigma_A^{Cm}\}$. *If both the conditions hold:*

C1 $\sigma_A \preceq_{\sigma_A^C} \sigma_A^m$ *and* $\{\sigma_A^m, \sigma_A^C\} \in \Gamma_A$ *and* $\mathcal{O}(\sigma_A^m) \subseteq \mathcal{O}(\sigma_A)$,

C2 $\sigma_A^C \preceq_{\sigma_A} \sigma_A^{Cm}$ *and* $\{\sigma_A, \sigma_A^{Cm}\} \in \Gamma_A$ *and* $\sigma_A^C, \sigma_A^{Cm}$ *are* σ_A, σ_A^m*-independent,*

then $\gamma_A \preceq \gamma_A^m$.

The proof constructs dominance preservation through domain-restricted strategy strengthening and epistemic separation, leveraging Lemma 4.2.11 to show preorder invariance under contextual modifications. See Kurpiewski et al. (2019b) for full details.

This result formalizes conditions for strategy replacement in distributed synthesis: component-wise strengthening preserves dominance when output containment and epistemic independence constraints are satisfied. The theorem enables decomposition-based optimization analogous to bisimulation quotients in strategic reasoning frameworks Belardinelli et al. (2017a).

4.2.4 Dominance-Based Depth-First Strategy Search: DominoDFS

The formal procedure for dominance-driven strategy synthesis is detailed in Algorithm 1. This recursive algorithm seeks to construct a strategy that satisfies p while extending σ_A from the initial state q_{init}. The algorithm operates under the invariant that σ_A remains p-free and initial state-loopless throughout execution.

Lines 1-8 implement a base case verification: when σ_A is empty, the synthesis process begins from q_{init}; otherwise, it identifies candidate states x where σ_A fails to enforce p (i.e., $x \notin V(p)$). If no such states exist, the current strategy is returned as a valid solution.

Algorithm 1 Dominance-Based Strategy Synthesis (**DominoDFS**$(\sigma_A, p, \preceq_{\mathcal{H}})$)

Input: σ_A - a p-free and initial state-loopless partial strategy for agent A
p - target proposition to enforce
$\preceq_{\mathcal{H}}$ - heuristic preorder for action prioritization
Output: Extension of σ_A enforcing p from initial state q_{init} or $\emptyset$ if none exists

1: **if** $\sigma_A = \emptyset$ **then**
2: $nxtstate \leftarrow q_{init}$ {Initialization}
3: **else if** there exists $x \in \mathcal{O}(\sigma_A)$ such that $x \notin V(p)$ **then**
4: $nxtstate \leftarrow x$ {Strategy not yet winning, extension required}
5: **else**
6:
7: **return** σ_A {Strategy already enforces p}
8: **end if**
9: $candactions \leftarrow d_A(nxtstate)$
10: $candactions.removeDominatedStrategies(\text{context} = \sigma_A)$
11: $candactions.removeLoops(\text{baseStrategy} = \sigma_A)$
12: **if** $candactions = \emptyset$ **then**
13:
14: **return** $\emptyset$ {No valid extensions available}
15: **end if**
16: $candactions.heuristicOrdering(\preceq_{\mathcal{H}})$
17: **for each** $actn \in candactions$ **do**
18: $solution \leftarrow \textbf{DominoDFS}(\sigma_A \cup \{(nxtstate, actn)\}, p, \preceq_{\mathcal{H}})$
19: **if** $solution \neq \emptyset$ **then**
20:
21: **return** $solution$
22: **end if**
23: **end for**
24:
25: **return** $\emptyset$ {No winning extension exists}

The algorithm then collects candidate actions *candactions* from $d_A(nxtstate)$ (Line 9), followed by two critical pruning operations: *removeDominatedStrategies* eliminates dominated actions relative to σ_A's context (Line 10), while *removeLoops* removes actions creating cycles (Line 11). Empty *candactions* after pruning indicates synthesis failure (Lines 12-14).

Remaining actions are ordered by the heuristic preorder $\preceq_{\mathcal{H}}$ (Line 16), with the algorithm recursively evaluating extensions (Lines 17-22). The search proceeds in $\preceq_{\mathcal{H}}$-priority order, terminating upon finding the first valid extension or exhausting all candidates. The implicit model structure follows the conventions established in prior sections for notational brevity.

4.3 Heuristics

While comparisons of input/output characteristics provide a sufficient condition for maintaining goal-enforceability, simpler strategies are often more practical in implementation. We formalize simplicity through domain-specific heuristic preorders $\precsim_{\text{comp}} \subseteq \Sigma_A \times \Sigma_A$, which may be naturally extended to partial strategies Γ_A. Three primary approaches emerge:

1. **Simple Reduction**: Relies solely on the preorder $\preceq$, prioritizing strategies with fewer actions and reduced branching factors.

2. **Epistemic Heuristic**: Compares cardinalities of exit state sets across partial strategies (considering indistinguishable states), preferring strategies with smaller exit sets ($|out(\sigma_A)| < |out(\sigma'_A)|$).

3. **Control Heuristic**: Evaluates the number of non-controlled states within exits, favoring strategies where exits contain fewer states not fully controlled by coalition A.

A state q is formally defined as *controlled* by coalition A if $d_{\mathbb{A}\text{gt}\setminus A}(q)$ constitutes a singleton set, indicating deterministic transition control by A.

4.4 Challenges and Lessons Learnt

The development of the forward-chaining strategy synthesis algorithm revealed critical challenges in handling epistemic classes and nondeterministic coalition dynamics. Strategic backtracking emerged as indispensable for reconciling locally winning strategies with global enforceability constraints, particularly when epistemic equivalence across nodes necessitated synchronized adjustments in parallel branches. This highlighted the inherent tension between local optimality and holistic strategy coherence, requiring explicit mechanisms to propagate constraints across epistemically indistinguishable states.

The vastness of the strategy space, even for models with hundreds of states, underscored the impracticality of exhaustive DFS. DominoDFS demonstrated that heuristic-driven prioritization (e.g., minimizing branching factors or exit states) could mitigate

combinatorial explosion, though trade-offs between computational efficiency and completeness remained. The co-NP-completeness of optimality verification (Theorem 4.2.7) further emphasized the need for approximations in real-world implementations.

The experimental evaluation of the proposed techniques across diverse benchmark models is presented in Chapter 9.

Key lessons included:

1. *Contextual independence* (Definition 4.2.10) enables decomposition of strategy synthesis into parallelizable components, mirroring bisimulation-based reductions in strategic logics.

2. *Input/output characteristics* (Definition 4.2.1) provide a robust framework for substituting partial strategies without compromising enforceability, as formalized in Theorem 60.

3. Heuristics prioritizing *output containment* (e.g., control heuristics favoring deterministic transitions) proved critical in balancing exploration depth and pruning efficacy.

These insights align with prior work in strategic dominance and epistemic model optimization Kurpiewski et al. (2019b), validating the theoretical underpinnings while exposing practical limitations in scaling nondeterministic coalition reasoning.

5

Strategy Optimization

Strategic optimization in multi-agent systems presents a critical challenge: balancing the complexity of decision-making frameworks with the preservation of functional guarantees. This chapter explores methodologies to simplify winning strategies while ensuring adherence to quality constraints, focusing on iterative refinement techniques that reconcile computational efficiency with strategic coherence. The discussion spans two primary paradigms: domination-based simplification and multi-criterial optimization, each addressing distinct yet interconnected facets of strategic design.

Section 5.1 generalizes the ideas presented in the previous chapters, and introduces a foundational optimization procedure rooted in domination principles, where strategies are restructured through ordered permutations of their components. This approach prioritizes maximal strategic configurations under heuristic preorders, enabling parallelizable simplification pathways. The theoretical underpinnings of this method emphasize scalability, with structural properties that suggest natural extensions to distributed implementations.

Section 5.2 expands the framework to dual-objective optimization, addressing competing goals such as outcome containment and action uniformity. The interplay between these criteria—often arising from perfect information strategies—demands a prioritized refinement hierarchy to resolve conflicts without compromising quality. The section formalizes this tension and proposes mechanisms to navigate trade-offs systematically.

Finally, Section 5.3 examines limitations in coalitional strategy synthesis, particularly the breakdown of epistemic closure properties when transitioning from individual to collective decision-making. Practical solutions are outlined to mitigate these chal-

Algorithm 2 Strategic Covering Optimization Procedure

Require: Initial strategy $\gamma_A \in \Gamma_A$, heuristic preorder $\preceq_{\mathcal{H}}$, quality predicate μ
Ensure: Optimized strategy $\gamma_A^m \in \Gamma_A$ satisfying $\gamma_A(\preceq \cap \preceq_{\mathcal{H}})\gamma_A^m$ and $\mu(\gamma_A^m)$

for each $\pi \in Sym(\gamma_A)$ **do**
  for each $1 \leq i \leq |\pi|$ **do**
    $\sigma_{A,i}^m$:=**SingleMax**$(\pi_i, \bigcup_{j \neq i} \pi_j, \preceq_{\mathcal{H}})$
    Update $\pi_i := \sigma_{A,i}^m$
  end for
  if $\mu(\pi)$ **then**
    return π
  end if
end for

lenges, balancing theoretical rigor with computational feasibility. The chapter concludes by highlighting opportunities for future work, including dynamic criterion weighting and coalition-aware refinements.

Throughout, the focus remains on anytime guarantees and pragmatic adaptability, ensuring utility even for models exceeding standard verification thresholds. The following sections elaborate on these paradigms, emphasizing modularity and theoretical consistency in strategic design.

The content of this chapter is based on the following papers: Jamroga and Kurpiewski (2023); Kurpiewski et al. (2019b).

5.1 Domination-Based Optimization

We evaluate our strategic optimization framework through iterative simplification of winning strategies while maintaining quality constraints. The procedure formalized in Algorithm 2 operates under quality predicate μ, where $\mu(\gamma_A^m)$ characterizes acceptable strategic configurations.

For any strategy $\gamma_A \in \Gamma_A$, let $Sym(\gamma_A)$ represent the set of all ordered rearrangements of its components. Strategic coverings γ_A^m may contain variable numbers of elements, with π_i denoting the i-th component in permutation π. Strategic permutations are treated as mutable vectors for computational purposes.

The optimization mechanism employs an auxiliary function **SingleMax**($\sigma'_A, \sigma^C_A, \preceq_{\mathcal{H}}$), which computes a maximal strategy σ^m_A satisfying $\{\sigma'_A, \sigma^C_A\} \preceq \cap \preceq_{\mathcal{H}} \{\sigma^m_A, \sigma^C_A\}$. This function's nondeterministic nature allows alternative optimization paths, while exhaustive implementations could explore complete maximal strategy spaces.

Proposition 5.1.1 (Parallelization Potential) *The outer loop iterations 1–9 are mutually independent and thus amenable to parallel execution. Under the conditions established in Theorem 4.2.12, the inner loop 2–5 supports concurrent processing of componentwise optimizations.*

These structural properties suggest promising avenues for distributed implementation optimizations, which we intend to investigate in subsequent work.

5.2 Multi-Criterial Domination

Unlike the single-criterion optimization in Section 5.1 focusing exclusively on outcome set tightness, our current approach introduces two orthogonal optimization dimensions: action uniformity and strategic tightness. This dual-objective formulation originates from the distinct nature of our initial strategic configurations, which derive from perfect information strategies rather than partial strategies. Consequently, our algorithm operates on strategic coverings that may exhibit multiple conflicts, necessitating simultaneous optimization across both dimensions to ensure effective simplification while preserving quality constraints.

5.2.1 Abstract Template

Given agent a's partial strategies σ_1, σ_2 defined over a shared epistemic class $[q]_a$ (i.e., $dom(\sigma_1), dom(\sigma_2) \subseteq [q]_a$ for some $q \in St$), we formalize a dual-criteria domination framework. The strategic space under consideration is denoted Σ, containing all relevant partial strategies.

Definition 5.2.1 ($(\mathcal{C}_1, \mathcal{C}_2)$-Domination Relation) *For criteria $\mathcal{C}_1, \mathcal{C}_2$ with associated total orders $\preceq_{\mathcal{C}_i}$ on Σ, define strict orders $\prec_{\mathcal{C}_i} = \preceq_{\mathcal{C}_i} \setminus (\preceq_{\mathcal{C}_i})^{-1}$. Strategy σ_2 $(\mathcal{C}_1, \mathcal{C}_2)$-dominates σ_1 iff:*

$$\sigma_1 \prec_{\mathcal{C}_1} \sigma_2 \quad \textit{and} \quad \sigma_1 \preceq_{\mathcal{C}_2} \sigma_2$$

This requires strict improvement on primary criterion $\mathcal{C}_1$ while maintaining non-inferiority on secondary criterion $\mathcal{C}_2$.

Definition 5.2.2 (Domination Hierarchy) *For $\sigma_2, \sigma_2' \in \Sigma$ both $(\mathcal{C}_1, \mathcal{C}_2)$-dominating σ_1, we say σ_2* better dominates *σ_1 iff $\sigma_2' \prec_{\mathcal{C}_1} \sigma_2$. Notably, this does not imply σ_2 dominates σ_2', as σ_2 might exhibit $\sigma_2' \prec_{\mathcal{C}_2} \sigma_2$.*

A strategy σ_2 best dominates *σ_1 under $(\mathcal{C}_1, \mathcal{C}_2)$ iff it dominates σ_1 and no other strategy in Σ better dominates σ_1. The set of best dominating strategies is denoted $Best_{\mathcal{C}_1,\mathcal{C}_2}(\sigma_1)$.*

5.2.2 Outcome- and Uniformity-Dominance

We formalize domination criteria over partial strategies σ_1, σ_2 sharing a common input domain $In \subseteq dom(\sigma_1), dom(\sigma_2)$. Let $Reach(In, \sigma_i)$ denote reachable states from In under σ_i, and define $RDom(In, \sigma_i) = dom(\sigma_i) \cap Reach(In, \sigma_i)$ as the *domain of relevance*—excluding unreachable states irrelevant for conflict analysis.

Definition 5.2.3 (Outcome Criterion $\mathcal{O}(In)$) *The outcome-based ordering $\sigma_1 \preceq_{\mathcal{O}(In)} \sigma_2$ holds iff $Reach(In, \sigma_2) \subseteq Reach(In, \sigma_1)$, i.e., σ_2 exhibits tighter outcome containment than σ_1.*

Definition 5.2.4 (Uniformity Criterion $\mathcal{U}(In)$) *For strategy σ_i, let $Conflicts(In, \sigma_i) = \{(q, q') \in Q^2 \mid \sigma_i(q) \neq \sigma_i(q')\}$ denote its conflict set over reachable states $Q = RDom(In, \sigma_i)$. Then $\sigma_1 \preceq_{\mathcal{U}(In)} \sigma_2$ iff $Conflicts(In, \sigma_2) \subseteq Conflicts(In, \sigma_1)$, requiring σ_2 to resolve at least as many conflicts as σ_1 within their relevant domains.*

Definition 5.2.5 (Bi-Criteria Domination) *A strategy σ_2:*

1. outcome-dominates *σ_1 iff $(\mathcal{O}(In), \mathcal{U}(In))$-domination holds*
2. uniform-dominates *σ_1 iff $(\mathcal{U}(In), \mathcal{O}(In))$-domination holds*

The concepts of better/best outcome-domination and uniform-domination follow directly from Definition 5.2.2.

5.2.3 Algorithm Overview

In this section, we propose an algorithm for strategy synthesis, as described below.

Proposition 5.2.6 (Prioritized Optimization Framework) *Our synthesis algorithm follows a two-phase approach:*

1. *First, construct a winning perfect information strategy $\widehat{\sigma}$ (if exists) through standard model checking procedures*

Algorithm 3 Strategic Synthesis Framework

Generate the model M
Construct a winning perfect information strategy $\widehat{\sigma}$
if $\widehat{\sigma}$ does not exist **then**
  return **false**
end if
Extract ordered information sets $\mathfrak{B} = \{(i, Q_i)\}_{i=1}^{n}$ from $M \dagger \widehat{\sigma}$
for $i = 1$ **to** n **do**
  Derive partial strategy $\sigma_i = \widehat{\sigma}|_{Q_i}$
  $In_i := Q_i \cap Reach\big(Reach(\mathcal{Q}_0, \widehat{\sigma}) \setminus Q_i,\ \widehat{\sigma} \setminus \sigma_i\big)$
  $RDom_i := Q_i \cap Reach(In_i, \sigma_i)$
  $Out_i := Reach(In_i, \sigma_i) \setminus Q_i$
  $Conflicts_i := Conflicts(RDom_i, \sigma_i)$
end for
Initialize partial strategy list $PStr \leftarrow \{\sigma_i\}_{i=1}^{n}$
return *PStr*

2. *Then, iteratively refine $\widehat{\sigma}$ using the dominance hierarchy from Section 5.2.1, prioritizing uniformity-dominance over outcome-dominance*

The process terminates at fixpoint (no further improvement) or upon timeout. The resulting strategy is validated for uniformity compliance before finalization.

This dual-phase execution ensures strategic simplification while maintaining quality constraints through its prioritized optimization sequence. The anytime property guarantees termination with a valid strategy *PStr* within finite computation bounds, even for models exceeding standard verification thresholds.

We will now define our procedure in more detail.

Definition 5.2.7 (Input Specification) *The algorithm accepts a triple $(M, q, \langle\langle a \rangle\rangle \mathbf{F}\varphi)$ as input:*

- *Model M with state space St*
- *Initial state $q \in St$*
- *Temporal formula $\langle\langle a \rangle\rangle \mathbf{F}\varphi$ requiring verification*

The initial epistemic class is defined as $\mathcal{Q}_0 = [q]_{\sim_a}$, representing all states indistinguishable to agent a at system initialization.

Algorithm 4 Iterative Bi-Criteria Optimization

repeat
  $OldPStr \leftarrow PStr$
  for $i = 1$ **to** n **do**
    repeat
      $OldPStr_i \leftarrow PStr(i)$
      if $\exists\sigma$ that uniform-best dominates $PStr(i)$ in In_i **then**
        Update $\sigma_i \leftarrow \sigma$ and recompute $RDom_i, Out_i, Conflicts_i$
      end if
      if $\exists\sigma$ that outcome-best dominates $PStr(i)$ in In_i **then**
        Update $\sigma_i \leftarrow \sigma$ and recompute $RDom_i, Out_i, Conflicts_i$
      end if
    until $PStr(i) = OldPStr_i$
  end for
  Reconstruct $\widehat{\sigma}$ from updated $PStr$
  for each $j \neq i$ **do**
    Update $In_j \leftarrow Q_j \cap Reach\big(Reach(\Omega_0, \widehat{\sigma}) \setminus Q_j,\ \widehat{\sigma} \setminus \sigma_j\big)$
  end for
until timeout or $PStr = OldPStr$
return $PStr$

Definition 5.2.8 (Data Structures) *The implementation utilizes three core data structures:*

1. Model representation M *with transition dynamics*
2. Information set hierarchy $\mathfrak{B} = \{(i, Q_i)\}_{i \in \mathbb{N}}$ *where each* Q_i *forms an equivalence class under* $\sim_a$
3. Partial strategy containers $(i, \sigma_i, In_i, RDom_i, Conflicts_i, Out_i)$ *with:*
   - i*: info set identifier*
   - σ_i*: action mapping restricted to* Q_i
   - In_i*: input states from* $Reach(\Omega_0, \widehat{\sigma}) \setminus Q_i$
   - $RDom_i$*: domain of relevance* $Q_i \cap Reach(In_i, \sigma_i)$
   - $Conflicts_i$*: conflict set* $\{(q, q') \in RDom_i^2 \mid \sigma_i(q) \neq \sigma_i(q')\}$
   - Out_i*: output states* $Reach(In_i, \sigma_i) \setminus Q_i$

The procedural core comprises Algorithms 3 (strategy synthesis) and 4 (iterative optimization). Algorithm 3 first attempts to construct a winning perfect information strategy $\widehat{\sigma}$ via model checking. If successful, it performs depth-first traversal of the pruned model $M \dagger \widehat{\sigma}$ to extract ordered information sets $\mathfrak{IS} = \{(i, Q_i)\}_{i=1}^{n}$, where each Q_i corresponds to an epistemic equivalence class under $\sim_a$. Subsequently, the algorithm decomposes $\widehat{\sigma}$ into partial strategies $\sigma_i = \widehat{\sigma}|_{Q_i}$, preserving the information set ordering.

Algorithm 4 executes cyclic optimization over partial strategies *PStr*:

1. Iterates through $\mathfrak{IS}$ in established order
2. Applies uniformity-dominance before outcome-dominance to each σ_i
3. Recomputes $In_i, RDom_i, Out_i, Conflicts_i$ after each update

The process terminates at convergence ($PStr = OldPStr$) or timeout. Strategic updates follow the prioritized dominance hierarchy from Section 5.2.1, ensuring quality preservation.

Proposition 5.2.9 (Anytime Strategic Refinement) *The procedure guarantees generation of a valid strategy PStr within bounded computational resources. This outperforms conventional methods Busard (2017); Busard et al. (2015); Jamroga et al. (2019a); Kurpiewski et al. (2019b); Lomuscio and Raimondi (2006); Lomuscio et al. (2017); Pilecki et al. (2017), which often fail to produce results for non-trivial models despite exhaustive resource allocation.*

5.3 Coalitional Strategies

Our current focus on individual strategies stems from fundamental limitations in coalition-based synthesis. The methodology naturally extends only to singleton coalitions due to the epistemic closure requirement: while information sets for single agents maintain indistinguishability closure, coalitional unions typically violate this property. Two approaches were considered:

Proposition 5.3.1 (Common Knowledge Closure) *Extending to coalitions via common knowledge neighborhoods $\mathcal{K}_A(q)$ would ensure epistemic closure, but practically leads to computational intractability as $\mathcal{K}_A(q)$ often equals the complete state space.*

Algorithm 5 Agent-Wise Strategic Refinement

OldPStr $\leftarrow$ *PStr*
for $i = 1$ **to** $|\mathfrak{I}|$ **do**
  repeat
    $OldPStr_i \leftarrow PStr(i)$
    if $\exists\sigma$ that uniform-best dominates $PStr(i)$ in In_i **then**
      Update $\sigma_i \leftarrow \sigma$ and recompute $RDom_i, Out_i, Conflicts_i$
    end if
    if $\exists\sigma$ that outcome-best dominates $PStr(i)$ in In_i **then**
      Update $\sigma_i \leftarrow \sigma$ and recompute $RDom_i, Out_i, Conflicts_i$
    end if
  until $PStr(i) = OldPStr_i$
end for
Reconstruct $\widehat{\sigma}$ from updated *PStr*
for each $j \neq i$ **do**
  Update $In_j \leftarrow Q_j \cap Reach\big(Reach(\Omega_0, \widehat{\sigma}) \setminus Q_j,\ \widehat{\sigma} \setminus \sigma_j\big)$
end for
return *PStr*

Proposition 5.3.2 (Alternating Optimization) *Algorithm 6 instead implements agent-wise optimization through alternating strategic refinement (Algorithm 5). This preserves the prioritized dominance hierarchy while enabling coalition analysis through iterative agent-specific improvements.*

The presented framework generalizes our synthesis methodology to arbitrary coalitions A, balancing epistemic requirements with computational feasibility through its alternating optimization structure.

5.4 Challenges and Lessons Learnt

The exploration of strategic optimization frameworks revealed critical limitations in extending our methodology to coalitional settings. A direct generalization of single-agent epistemic closure properties to multi-agent coalitions A proved infeasible due to the breakdown of indistinguishability guarantees. Specifically, unions of individual information sets $Q_i \cup Q_j$ (for agents $a_i, a_j \in A$) often fail to maintain the epistemic

Algorithm 6 Coalitional Optimization Framework

```
repeat
  OldPStr ← PStr
  for each a ∈ A do
    PStr ← optimize_once(PStr)
  end for
until timeout or PStr = OldPStr
return PStr
```

closure required for conflict resolution, rendering traditional coalition-based synthesis approaches ineffective.

Two key challenges emerged during this investigation:

1. **Computational Intractability in Common Knowledge Closure**: Proposition 5.3.1 demonstrates that enforcing epistemic closure through common knowledge neighborhoods $K_A(q)$ typically collapses to the full state space St, eliminating the practical benefits of partial information strategies. This aligns with findings in distributed verification literature where global knowledge assumptions lead to exponential complexity blowups.

2. **Conflict Between Dominance Criteria**: The dual-objective optimization framework (Definition 5.2.5) introduces tension between outcome tightness and action uniformity. While prioritizing uniformity-dominance (Proposition 5.2.6) resolves this hierarchy, it requires careful balancing to avoid sacrificing strategic effectiveness for simplification gains.

Our resolution to these challenges yielded three significant lessons:

- **Alternating Optimization Enables Coalition Scalability**: Algorithm's 6 agent-wise refinement loop (Algorithm 5) circumvents epistemic closure violations by treating coalitions as iterative consensus builders rather than unified decision-makers. This approach maintains computational feasibility while approximating coalitional behavior.

- **Anytime Properties as Practical Guarantees**: The framework's anytime nature (Proposition 5.2.9) ensures usability even for models exceeding verification

thresholds. This contrasts sharply with conventional methods that fail entirely when resource bounds are exceeded.

- **Strategic Trade-offs Are Inevitable**: The dominance hierarchy formalized in Definition 5.2.2 forces explicit prioritization of optimization criteria. In practice, this revealed that outcome-dominance often requires reintroducing complexity that uniform-dominance had previously eliminated, highlighting the need for domain-specific weighting of C_1 and C_2.

This chapter underscores the necessity of pragmatic approximations in strategic synthesis, particularly when reconciling theoretical epistemic requirements with real-world computational constraints. Future work will explore dynamic criterion weighting and hybrid symbolic-explicit model representations to further bridge this gap.

6

Reducing the State-Space

Model checking multi-agent systems often confronts the computational burden of state-space explosion, where the exponential growth of possible system states renders exhaustive verification infeasible. This chapter explores methodologies to mitigate such complexity through abstractions and reductions that balance scalability with logical fidelity. By simplifying the representation of states, actions, and transitions, these techniques aim to preserve critical strategic and temporal properties while discarding redundant or non-essential details.

The chapter is structured into three core parts. First, we examine abstraction frameworks that cluster concrete states and actions into equivalence classes, enabling the derivation of abstract models that approximate strategic capabilities of agents or coalitions. Second, we investigate partial-order reduction (POR) strategies, which exploit independence and invisibility of transitions to dynamically prune the state space without compromising verification outcomes. Finally, we conclude by pointing out encountered challenges and lessons learnt.

A central theme is the trade-off between precision and computational tractability. While abstractions usually introduce approximation, their formal guarantees ensure that verification results remain sound within bounded contexts. Similarly, POR techniques leverage structural properties of systems to avoid redundant exploration of equivalent execution paths. Through theoretical analysis and domain-specific applications, this chapter illustrates how these methods address complexity while maintaining rigorous alignment with the original system's behavior.

The content of this chapter is based on the following papers: Jamroga et al. (2019a, 2022b).

6.1 Combining Fixpoint Approximation and Abstraction

Abstraction Clarke et al. (1994); Cousot and Cousot (1977) constitutes a pivotal technique for mitigating computational complexity in model checking by reducing the state space through state and action clustering. This methodology partitions the concrete model's states into equivalence classes that define the abstract model's state space, while concurrently aggregating analogous concrete actions into abstract counterparts. Within temporal logic verification, may/must abstractions Bruns and Godefroid (1999); Godefroid and Jagadeesan (2002) employing three-valued semantics demonstrate particular efficacy: may abstractions preserve existential path quantifiers while systematically underapproximating universal properties by introducing additional execution paths, whereas must abstractions conversely maintain universal properties at the expense of existential overapproximation. Both approaches, while capable of yielding definitive verification outcomes under favorable conditions, typically produce bounded approximations that may remain inconclusive.

The conceptual symmetry between these model transformation techniques and fixpoint approximations (Chapter 3) reveals an opportunity for synergistic integration. We formalize a novel abstraction framework for $\mathbf{ATL}_{\mathrm{ir}}$ inspired by three-valued abstraction methodologies for strategic reasoning in concurrent games B. and Lomuscio (2017); Ball and Kupferman (2006), and demonstrate its compatibility with existing fixpoint approximation strategies. This dual-approximation paradigm enables simultaneous model and formula transformation while maintaining consistent bounding properties, culminating in an enhanced verification methodology subjected to rigorous empirical evaluation.

6.1.1 State and Action Abstraction for $\mathbf{ATL}_{\mathrm{ir}}$

We present a semi-formal characterization of the abstraction framework for strategic reasoning, comprising two dual constructions: the *lower abstraction* $\mathcal{A}^L_A(M)$ and *upper abstraction* $\mathcal{A}^U_A(M)$ of a model M with respect to agent coalition A. These abstractions systematically underapproximate and overapproximate the strategic capabilities of A

respectively, while eliminating non-coalition agents through nondeterministic encapsulation of their behaviors.

The abstractions are derived from an *abstraction generator* $\mathcal{A} = (\mathcal{A}_S, \mathcal{A}_{Ac})$, where:

- $\mathcal{A}_S : St \to \mathcal{A}_A^L(St)$ partitions concrete states into abstract equivalence classes
- $\mathcal{A}_{Ac} : Act \to \mathcal{A}_A^L(Act)$ defines action renaming/mapping between state spaces

Lower Abstraction ($\mathcal{A}_A^L(M)$):

- *Valuation*: Proposition p holds in abstract state $\widehat{q}$ iff p holds in *all* $q' \in \widehat{q}$ (pessimistic interpretation)
- *Observational Equivalence*: Abstract states $\widehat{q}_1 \sim \widehat{q}_2$ iff $\exists q_1' \in \widehat{q}_1, \exists q_2' \in \widehat{q}_2$ with $q_1' \sim q_2'$ in M
- *Protocol Construction*: For each $\widehat{q}$, collect actions present in *all* protocols of concrete states $q' \in \bigcup_{\widehat{q'} \sim \widehat{q}} \widehat{q'}$, then apply $\mathcal{A}_{Ac}$
- *Transition Semantics*: $\widehat{q_2} \in \mathcal{A}_A^L(M)$ satisfies $\widehat{q_2} = \widehat{o}(\widehat{q_1}, \widehat{\alpha}_1, \ldots, \widehat{\alpha}_{|A|})$ iff $\exists q_1 \in \widehat{q_1}, \exists q_2 \in \widehat{q_2}$ in M with $q_2 = o(q_1, \alpha_1, \ldots, \alpha_{|A|})$, establishing existential path quantification

Upper Abstraction ($\mathcal{A}_A^U(M)$):

- *Valuation*: Proposition p holds in $\widehat{q}$ iff p holds in *some* $q' \in \widehat{q}$ (optimistic interpretation)
- *Observational Equivalence*: $\widehat{q}_1 \sim \widehat{q}_2$ iff $\forall q_1' \in \widehat{q}_1, \forall q_2' \in \widehat{q}_2$ we have $q_1' \sim q_2'$ in M
- *Protocol Construction*: For each $\widehat{q}$, aggregate actions from *any* protocol of concrete states $q' \in \bigcup_{\widehat{q'} \sim \widehat{q}} \widehat{q'}$, then apply $\mathcal{A}_{Ac}$
- *Transition Semantics*: $\widehat{q_2} = \widehat{o}(\widehat{q_1}, \widehat{\alpha}_1, \ldots, \widehat{\alpha}_{|A|})$ in $\mathcal{A}_A^U(M)$ iff $\forall q_1 \in \widehat{q_1}, \exists q_2 \in \widehat{q_2}$ in M such that $q_2 = o(q_1, \alpha_1, \ldots, \alpha_{|A|})$ after renaming, enforcing universal path quantification

This framework systematically preserves the fundamental approximation properties of may/must abstractions while extending them to strategic reasoning contexts, with formal guarantees established in Section 6.1.4 through a representative example.

6.1.2 Approximation Semantics for $\mathbf{ATL}_{\text{ir}}$ in Abstract Models

We formalize the abstraction framework's transformation of iCGS models to establish systematic underapproximation and overapproximation guarantees for strategic abilities. For any coalition A, the lower abstraction $\mathcal{A}_A^L(M)$ conservatively restricts strategic capabilities by enforcing universal satisfaction of atomic propositions and existential path preservation, while the upper abstraction $\mathcal{A}_A^U(M)$ expansively generalizes abilities through optimistic proposition valuation and universal path inclusion. This ensures:

$$\mathcal{A}_A^L(M), q \models \langle\langle A \rangle\rangle \gamma \Rightarrow M, q \models \langle\langle A \rangle\rangle \gamma \Rightarrow \mathcal{A}_A^U(M), q \models \langle\langle A \rangle\rangle \gamma$$

for formulae $\langle\langle A \rangle\rangle \gamma$ without nested strategic modalities.

The extension to full $\mathbf{ATL}_{\text{ir}}$ requires careful treatment of negation through *strategic tagging*. We recursively annotate subformulae with superscripts L or U, determining their evaluation context:

- Strategic modalities $\langle\langle A \rangle\rangle_{\text{ir}}$ inherit tagging direction: $\langle\langle A \rangle\rangle_{\text{ir}}^L$ in $\mathcal{A}_A^L(M)$, $\langle\langle B \rangle\rangle_{\text{ir}}^U$ in $\mathcal{A}_A^U(M)$
- Atomic propositions p receive independent tags: p_1^U for optimistic valuation, p_2^L for pessimistic valuation
- Negation flips tags: $\neg\varphi^L \equiv \neg\varphi^U$, $\neg\varphi^U \equiv \neg\varphi^L$

This mechanism enables hybrid evaluation of nested modalities. For instance:

$$\langle\langle A \rangle\rangle_{\text{ir}}^L \mathbf{F} \neg \langle\langle B \rangle\rangle_{\text{ir}}^U \mathbf{G}(\mathsf{p}_1^U \wedge \neg \mathsf{p}_2^L)$$

demonstrates how alternating approximations maintain sound bounds through tag propagation. Atomic propositions remain agnostic to coalition-specific abstractions, permitting interpretation in $\mathcal{A}_\emptyset^U(M)$ and $\mathcal{A}_\emptyset^L(M)$ without loss of generality.

The lower abstraction tagging $TR_L(\cdot)$ is defined through recursive transformation

of $\mathbf{ATL}_{\text{ir}}$ formulae:

$$
\begin{aligned}
TR_L(p) &= p^L && \text{(pessimistic proposition valuation)}\\
TR_L(\langle\!\langle A\rangle\!\rangle_{\text{ir}}\varphi) &= \langle\!\langle A\rangle\!\rangle_{\text{ir}}^L TR_L(\varphi) && \text{(conservative strategic capability)}\\
TR_L(\neg\varphi) &= \neg TR_U(\varphi) && \text{(negation-induced approximation flip)}\\
TR_L(\varphi \wedge \psi) &= TR_L(\varphi) \wedge TR_L(\psi) && \text{(conjunction preservation)}\\
TR_L(\mathbf{X}\varphi) &= \mathbf{X}\, TR_L(\varphi) && \text{(temporal operator compatibility)}\\
TR_L(\mathbf{G}\varphi) &= \mathbf{G}\, TR_L(\varphi) && \text{(universal path quantification)}\\
TR_L(\psi\, \mathbf{U}\, \varphi) &= TR_L(\psi)\, \mathbf{U}\, TR_L(\varphi) && \text{(fixed-point operator consistency)}
\end{aligned}
$$

This recursive scheme ensures systematic approximation bounds by maintaining monotonicity across logical operators. The corresponding upper abstraction tagging $TR_U(\cdot)$ follows symmetric rules with dual valuation principles, enabling rigorous hybrid evaluation of arbitrarily nested strategic formulae.

The upper abstraction tagging $TR_U(\cdot)$ follows a symmetric recursive structure with dual valuation principles:

$$
\begin{aligned}
TR_U(p) &= p^U && \text{(optimistic proposition valuation)}\\
TR_U(\langle\!\langle A\rangle\!\rangle_{\text{ir}}\varphi) &= \langle\!\langle A\rangle\!\rangle_{\text{ir}}^U TR_U(\varphi) && \text{(expansive strategic capability)}\\
TR_U(\neg\varphi) &= \neg TR_L(\varphi) && \text{(negation-induced approximation flip)}\\
TR_U(\varphi \wedge \psi) &= TR_U(\varphi) \wedge TR_U(\psi) && \text{(conjunction preservation)}\\
TR_U(\mathbf{X}\varphi) &= \mathbf{X}\, TR_U(\varphi) && \text{(temporal operator compatibility)}\\
TR_U(\mathbf{G}\varphi) &= \mathbf{G}\, TR_U(\varphi) && \text{(universal path quantification)}\\
TR_U(\psi\, \mathbf{U}\, \varphi) &= TR_U(\psi)\, \mathbf{U}\, TR_U(\varphi) && \text{(fixed-point operator consistency)}
\end{aligned}
$$

This dual tagging mechanism completes the formalization of three-valued strategic reasoning, ensuring complementary approximation bounds through mutual recursion between $TR_L(\cdot)$ and $TR_U(\cdot)$. The framework guarantees sound hybrid evaluation of arbitrary $\mathbf{ATL}_{\text{ir}}$ formulae with precise quantification of strategic abilities.

We formalize the three-valued semantics through dual satisfaction relations $\models^L$ and $\models^U$, defined over the family of abstracted models $M_A^x = \mathcal{AG}(x, A)(M)$ for all $x \in \{L, U\}$ and $A \subseteq \mathbb{A}\text{gt}$. For tagged formulae φ^x, the semantical clauses extend the standard $\mathbf{ATL}_{\text{ir}}$ interpretation (Section 2.1.2) as follows:

$$M_A^y, \widehat{q} \models p^x \quad \Leftrightarrow M_A^x, \widehat{q} \models p \quad \text{(proposition tagging consistency)}$$
$$M_A^y, \widehat{q} \models \langle\!\langle B \rangle\!\rangle_{\text{ir}}^x \varphi \quad \Leftrightarrow M_B^x, \widehat{q} \models \langle\!\langle B \rangle\!\rangle_{\text{ir}} \varphi \quad \text{(strategic modality projection)}$$

This formulation ensures that:

1. The lower abstraction M_A^L enforces universal quantification over concrete states for proposition verification
2. The upper abstraction M_A^U applies existential quantification for proposition satisfaction
3. Strategic modalities maintain approximation directionality through explicit model projection
4. Tag propagation preserves logical structure across Boolean and temporal operators

The interplay between abstraction transformers $\mathcal{AG}(x, A)$ and tagged formulae $\cdot^x$ establishes a rigorous framework for bounded strategic reasoning with guaranteed soundness properties.

We formalize the bounded satisfaction relations for three-valued strategic reasoning:

$$M, q \models^L \varphi \quad \Leftrightarrow M_\emptyset^L, \mathcal{A}_S(q) \models TR_L(\varphi) \quad \text{(lower abstraction semantics)}$$
$$M, q \models^U \varphi \quad \Leftrightarrow M_\emptyset^U, \mathcal{A}_S(q) \models TR_U(\varphi) \quad \text{(upper abstraction semantics)}$$

This formulation establishes strict correspondence between concrete model satisfaction and abstracted evaluations, where $\mathcal{A}_S$ denotes the state abstraction mapping induced by $\mathcal{A}$.

Theorem 6.1.1 (Approximation Soundness Jamroga et al. (2019a)) *For any* $\mathbf{ATL}_{\text{ir}}$ *formula* φ *and state* q *in iCGS* M*:*

$$M, q \models^L \varphi \quad \Rightarrow \quad M, q \models \varphi \quad \Rightarrow \quad M, q \models^U \varphi$$

This theorem guarantees that the lower abstraction provides a sufficient condition for formula satisfaction, while the upper abstraction establishes a necessary condition. The proof follows from the preservation of logical structure under abstraction transformers and is detailed in Jamroga et al. (2019a).

6.1.3 Generalized Approximation Semantics: Combining Approximation on Models and Formulae

We have presented two methods of approximating the output of model checking for formulae of $\mathbf{ATL}_{\text{ir}}$. One transforms formulae into ones that provide a lower and an upper bound on the truth value of the original formula. The other one, discussed in the preceding subsections, produces analogous transformations of models. Here, we point out that both methods can be combined in a straightforward manner. We formalize the integration of formula- and model-based approximations through extended strategic tagging mechanisms. The lower abstraction tagging $TR_L(\cdot)$ now incorporates:

$$\begin{aligned} TR_L(\langle A\rangle^{\bullet}\varphi) &= \langle A\rangle^{\bullet L}\, TR_L(\varphi) && \text{(conservative steadfast operator)}\\ TR_L(Z) &= Z && \text{(fixpoint variable preservation)}\\ TR_L(\mu Z.\varphi) &= \mu Z.\, TR_L(\varphi) && \text{(monotonic fixpoint distribution)}\\ TR_L(\nu Z.\varphi) &= \nu Z.\, TR_L(\varphi) && \text{(co-monotonic fixpoint distribution)} \end{aligned}$$

with symmetric definitions for $TR_U(\cdot)$. The semantical interpretation extends to:

- $M_A^y, \widehat{q} \models \langle B\rangle^{\bullet x}\varphi \Leftrightarrow M_B^x, \widehat{q} \models \langle B\rangle^{\bullet}\varphi$ (strategic steadfast consistency)

This hybrid framework systematically preserves approximation bounds across arbitrary nested strategic modalities and fixed-point constructs, enabling seamless integration with $\mathbf{AE}\mu\mathbf{C}$ verification procedures.

Now, the result of Theorem 6.1.1 can be extended to the combination of abstraction and fixpoint approximation:

Theorem 6.1.2 (Combined Approximation Soundness Jamroga et al. (2019a)) *For any* $\mathbf{ATL}_{\text{ir}}$ *formula* φ *and state* q *in iCGS* M*, the hybrid approximation framework guarantees:*

$$M, q \models^{L} TR_L(\varphi) \quad \Rightarrow \quad M, q \models \varphi \quad \Rightarrow \quad M, q \models^{U} TR_U(\varphi)$$

This result demonstrates that simultaneous formula transformation and model abstraction preserve the stratified approximation bounds established in Theorem 6.1.1. The lower abstraction $TR_L(\varphi)$ provides a sufficient condition for φ to hold in the concrete model, while the upper abstraction $TR_U(\varphi)$ establishes a necessary condition. The

interplay between fixpoint approximation and strategic tagging ensures rigorous soundness across arbitrary combinations of nested modalities and epistemic constraints. The proof can be found in Jamroga et al. (2019a).

6.1.4 Lower and Upper Abstractions for the Bridge Model

A natural abstraction commonly employed by human players involves disregarding the ranks of cards below a specified threshold R. Under this framework, players retain explicit knowledge of both rank and suit for cards at or above R, while only suit information is preserved for cards below R. For instance, when $R = J$ and a player holds $\spadesuit AK92\ \diamondsuit 10\ \clubsuit J653$, the abstraction yields $\spadesuit AKxx\ \diamondsuit x\ \clubsuit Jxxx$, where x denotes a low card of indeterminate rank. Similarly, an opponent's play of $\diamondsuit 8$ would be represented as $\diamondsuit x$ in the abstracted model. This methodology establishes a scalable family of abstractions for any bridge endplay model M, enabling dynamic adjustment of granularity through parameter r.

Formally, the abstraction is defined via a family of deck simplifiers $simpl_r : \text{Deck}_n \to \text{Deck}_r$ for $1 \leq r \leq n$, given by:

$$simpl_r((\text{rank}, \text{suit})) = \begin{cases} (\text{rank}, \text{suit}) & \text{if rank} > n - r + 1, \\ (n - r + 1, \text{suit}) & \text{otherwise.} \end{cases}$$

Here, the top $r - 1$ ranks are preserved, while lower ranks are clustered into a single equivalence class. For $r = 1$, all ranks collapse to a single class; for $r = n$, the original deck structure remains unchanged.

The abstraction generator $\mathcal{A}^r = (\mathcal{A}^r_S, \mathcal{A}^r_{Ac})$ operates as follows:

- $\mathcal{A}^r_S((hands, tricks, next, board, lead, history, clock, suit)) =$
  $(simpl_r(hands), tricks, next, simpl_r(board), lead, simpl_r(history), clock, suit)$,
- $\mathcal{A}^r_{Ac}(c) = simpl_r(c)$ for card actions c,
- $\mathcal{A}^r_{Ac}(\text{wait}) = \text{wait}$ for non-card actions.

This formulation ensures consistency between state and action abstractions while maintaining computational flexibility.

6.2 Partial-Order Reductions

Partial order reduction (POR) is a state space minimization technique originating nearly three decades ago. The method dynamically selects a representative (i.e., provably sufficient) subset of enabled transitions during system unfolding, grounded in transition equivalence relations. This approach avoids explicit construction of the potentially intractable full state space model, enabling scalable verification.

POR frameworks have been rigorously developed for various Linear Temporal Logic (**LTL**) and Computation Tree Logic (**CTL**) variants, including temporal-epistemic extensions. A significant advancement demonstrated in Jamroga et al. (2018b) extended the **LTL**-based reduction mechanism to the strategic logic **sATL*** without altering computational complexity, thereby broadening its applicability to multi-agent system verification. Subsequent work Jamroga et al. (2020a) confirmed the adaptation's validity under the revised AMAS execution semantics, which we formally incorporate in our theoretical framework.

6.2.1 Conceptual Machinery

We first recall the concept of stuttering equivalence. Intuitively, two paths are stuttering equivalent if they can be divided into corresponding finite segments, each satisfying exactly the same propositions. If all states in corresponding segments are also indistinguishable for agents $i \in J$, then we say the paths are $J-$ stuttering equivalent.

Definition 6.2.1 ($(J$-$)$stuttering equivalence) *Paths $\pi, \pi' \in \Pi_M(q)$ are* stuttering equivalent*, denoted $\pi \equiv_s \pi'$, if there exists a partition $B_0 = (\pi[0], \dots, \pi[i_1 - 1])$, $B_1 = (\pi[i_1], \dots, \pi[i_2 - 1])$, ... of the states of π, and an analogous partition $B'_0, B'_1, \dots$ of the states of π', s.t. for each $j \geq 0 : B_j$ and B'_j are nonempty and finite, and $V(q) \cap PV = V(q') \cap PV$ for every $q \in B_j$ and $q' \in B'_j$.*

If $\pi \equiv_s \pi'$, and additionally it holds that $\forall j > 0 \quad \forall q \in B_j, q' \in B'_j : \quad q \sim_J q'$, then paths π and π' are J-stuttering equivalent*, denoted $\pi \equiv_s^J \pi'$.*

States q and q' are stuttering path equivalent *(resp.* J-stuttering path equivalent*), denoted $q \equiv_s q'$ (resp. $q \equiv_s^J q'$), iff for every path π starting from q, there is a path π' starting from q' such that $\pi' \equiv_s \pi$ (resp. $\pi' \equiv_s^J \pi$), and for every path π' starting from q', there is a path π starting from q such that $\pi \equiv_s \pi'$ (resp. $\pi \equiv_s^J \pi'$).*

Models M and M' are stuttering path equivalent *(resp.* J-stuttering path equivalent*), denoted $M \equiv_s M'$ (resp. $M \equiv_s^J M'$), iff they have the same initial states, and*

for each initial state $\iota_i \in I$ and each path $\pi \in \Pi_M(\iota_i)$, there is a path $\pi' \in \Pi_{M'}(\iota_i)$ such that $\pi \equiv_s \pi'$ (resp. $\pi \equiv_s^J \pi'$).

The POR algorithm uses the notions of invisible and independent events. Intuitively, an event is invisible iff it does not change the valuations of the propositions. Two events are independent iff at least one is invisible and they are not in the same agent's repertoire. We designate a subset of agents $A \subseteq$ A whose events are visible by definition.

Definition 6.2.2 (Invisibility and independence of events) *Let $M = IIS(S, I)$, and $A \subseteq$ A. An event $\alpha \in Evt$ is* invisible *wrt. A and PV if $Agent(\alpha) \cap A = \emptyset$ and for each two global states $q, q' \in St$ we have that $q \xrightarrow{\alpha} q'$ implies $V(q) \cap PV = V(q') \cap PV$. The set of all invisible events for A, PV is denoted by $Invis_{A,PV}$, and its closure, i.e. the set of visible events, by $Vis_{A,PV} = Evt \setminus Invis_{A,PV}$.*

The notion of independence *$Ind_{A,PV} \subseteq Evt \times Evt$ is defined as: $Ind_{A,PV} = \{(\alpha, \alpha') \in Evt \times Evt \mid Agent(\alpha) \cap Agent(\alpha') = \emptyset\} \setminus (Vis_{A,PV} \times Vis_{A,PV})$. Events $\alpha, \alpha' \in Evt$ are called* dependent *if $(\alpha, \alpha') \notin Ind_{A,PV}$. If it is clear from the context, we omit the subscript PV.*

The reduced model (or *submodel*) $M' \subseteq M$ obtained with POR extends the same AMAS S as $M = IIS(S, I)$. In particular, we have $St' \subseteq St$, $I = I'$, T is an extension of T', and $V' = V|_{St'}$. Note that, for each $q \in St'$, it holds that $\Pi_{M'}(q) \subseteq \Pi_M(q)$.

M' is generated by modifying the standard DFS,so that for each q, the successor state q_1 such that $q \xrightarrow{\alpha} q_1$ is selected from $E(q) \cup \{\epsilon\}$ such that $E(q) \subseteq enabled(q) \setminus \{\epsilon\}$. That is, the algorithm always selects ϵ, plus a subset of the enabled events at q. This modified DFS is called for each initial state of the model, and we have $\Pi_{M'} = \bigcup_{q \in I} \Pi_{M'}(q)$. The conditions on the heuristic selection of $E(q)$ given below are inspired by Jamroga et al. (2018b).

C1 Along each path π in M that starts at g, each event that is dependent on an event in $E(q)$ cannot be executed in π without an event in $E(q)$ being executed first in π. Formally, $\forall \pi \in \Pi_M(q)$ such that $\pi = q_0\alpha_0q_1\alpha_1 \ldots$ with $q_0 = q$, and $\forall b \in Evt$ such that $(b, c) \notin Ind_A$ for some $c \in E(q)$, if $\alpha_i = b$ for some $i \geq 0$, then $\alpha_j \in E(q)$ for some $j < i$.

C2 If $E(q) \neq enabled(q) \setminus \{\epsilon\}$, then $E(q) \subseteq Invis_A$.

C3 For every cycle in M' containing no ϵ-transitions, there is at least one node q in the cycle for which $E(q) = enabled(q) \setminus \{\epsilon\}$, i.e., all the successors of q are expanded.

Submodel $M' \subseteq M$ generated with this algorithm satisfies property $_A$ Jamroga et al. (2020a):

$$\forall \sigma_A \in \Sigma_A^{\mathrm{ir}}\, \forall \iota_i \in I\, \forall \pi \in out_M(\iota_i, \sigma_A)\, \exists \pi' \in out_{M'}(\iota_i, \sigma_A)\, \pi \equiv_s \pi'.$$

6.2.2 POR for sATL*K

We observe that the algorithm for **sATL*** Jamroga et al. (2018b, 2020a) can be applied also to formulae that include the knowledge operator (in subformulae of the form $\mathrm{K}i\varphi$), provided that $J \subseteq A$. That is, any agents in these epistemic subformulae are added to the set $A \subseteq \mathrm{A}$ that parametrises the relations of invisibility and independence.

Theorem 6.2.3 (Jamroga et al. (2022b)) *Let $J \subseteq A \subseteq \mathrm{A}$, $M = IIS(S, I)$, and let $M' \subseteq M$ be the reduced model generated by DFS with the choice of $E(q')$ for $q' \in St'$ given by conditions* **C1-C3**. *Then, for any starting state $\iota_i \in I$ and any* **sATL*K** *formula φ over PV that refers only to coalitions $\hat{A} \subseteq A$, we have that $M, \iota_i \models \varphi$ iff $M', \iota_i \models \varphi$.*

6.2.3 POR for "Subjective" Semantics of Ability

Moreover, the reduction scheme remains applicable in the subjective semantics of strategic ability.

Theorem 6.2.4 (Jamroga et al. (2022b)) *Let $\hat{A} \subseteq A$ and $I_{\hat{A}} = I \cup \{q \in St \mid \exists_{\iota_i \in I} : q \sim_{\hat{A}} \iota_i\}$. That is, the set of initial states is comprised of previously designated subset $\iota \subseteq St$, plus all states indistinguishable from those in ι for agents in coalition $\hat{A}$. Let $M = IIS(S, I_{\hat{A}})$, and let M' be the reduction of M generated by POR using conditions* ***C1-C3*** *for the choice of ample sets. Then, for any starting state $\iota_i \in I_{\hat{A}}$ and any* **sATL*K** *formula φ over PV that refers only to coalition $\hat{A}$, we have that $M, \iota_i \models_S \varphi$ iff $M', \iota_i \models_S \varphi$.*

6.3 Challenges and Lessons Learnt

The integration of fixpoint approximation and three-valued abstraction for strategic reasoning in ATLir introduced multifaceted challenges that required both theoretical rigor and practical ingenuity. This section synthesizes the obstacles encountered during

the framework's development and its empirical validation across domains like Bridge endplay analysis, alongside the critical insights gained. By dissecting these challenges, we uncover the strengths and limitations of hybrid verification methodologies that combine model reductions with formula transformations, offering actionable strategies for mitigating state-space complexity in multi-agent systems.

The experimental evaluation of the proposed techniques across diverse benchmark models is presented in Chapter 9.

Challenges

- **Preservation of Logical Structure**: Maintaining monotonicity and duality across Boolean/temporal operators while integrating fixpoint approximations with three-valued abstraction required rigorous formalization to avoid logical inconsistencies (as highlighted in Theorem 6.1.2).
- **Scalability of Hybrid Evaluation**: Simultaneous model and formula transformation introduced computational overhead, particularly in handling nested strategic modalities with alternating tags $\mathcal{T}^L$ and $\mathcal{T}^U$.
- **Abstraction Granularity vs. Precision**: The Bridge model abstraction (Section 6.1.4) revealed a trade-off between reducing state-space complexity via $\texttt{simpl}_r$ and retaining critical strategic information (e.g., card ranks above threshold R).
- **Negation Handling in Strategic Tagging**: Ensuring sound propagation of negation-induced tag flips ($\neg\varphi^L \equiv \neg\varphi^U$) across arbitrary nesting of modalities demanded careful alignment with the three-valued semantics.
- **Dynamic Coalition Adaptation**: Extending partial-order reductions (POR) to subjective semantics of ability (Section 6.2.3) required redefining stuttering equivalence for indistinguishable states under coalition-specific observational equivalence.

Lessons Learnt

- **Synergy of Dual Approximations**: The combination of fixpoint-based formula approximations and abstraction-based model reductions proved more effective than standalone methods, as demonstrated by the stratified bounds in Theorem 6.1.1 and Theorem 6.1.2.

- **Tagged Formulae as a Unifying Mechanism**: Strategic tagging ($\mathcal{T}^L, \mathcal{T}^U$) enabled systematic hybrid evaluation of $\mathbf{ATL}_{\text{ir}}$ formulae, preserving approximation guarantees even with nested epistemic constraints.
- **Agent-Centric POR Limitations**: While POR preserved $\mathcal{J}$-stuttering equivalence (Definition 6.2.1), its applicability to multi-agent systems hinged on restricting visible events to coalitions of interest ($\mathcal{A} \subseteq \mathcal{A}$), as formalized in Theorem 6.2.3.
- **Formal Guarantees Require Contextual Adaptation**: Theoretical soundness of abstractions (e.g., M_A^L, M_A^U) could not be universally applied without domain-specific adjustments to the abstraction mappings ($\mathcal{A}^S, \mathcal{A}^{Ac}$).

Part II

Practical Verification: Models, Implementation and Experimental Evaluation

7

Modelling the Real World

The formal verification process, particularly in domains requiring critical property validation such as receipt-freeness in electronic voting protocols, necessitates two fundamental components: (1) precise logical formulation of the target property and (2) rigorous specification of the computational model. While temporal logic formula construction provides a formal framework for property verification, model generation constitutes a more intricate challenge due to the absence of automated translation mechanisms from system specifications to formal computational models like concurrent game structures (**CGS**). This manual specification process typically requires domain expertise and careful abstraction of system components.

Key considerations in model design include:

- *Abstraction adequacy*: Determining whether the model captures essential agent behaviors, interaction protocols, and epistemic states relevant to the verification task.
- *Semantic fidelity*: Ensuring accurate representation of system constraints and operational parameters.
- *Computational tractability*: Balancing model complexity against verification algorithm performance to avoid state-space explosion issues.

These challenges highlight the inherent tension between model expressiveness and analytical feasibility. Over-abstraction risks omitting security-critical behaviors, while excessive detail can lead to intractable verification problems manifesting as timeouts

or memory exhaustion during model checking. This chapter subsequently examines formal modeling methodologies that effectively address these competing requirements in real-world verification scenarios.

The content of this chapter is based on the following papers: Jamroga and Kurpiewski (2023); Jamroga et al. (2019a, 2022a,b); Kurpiewski and Marmsoler (2019); Kurpiewski et al. (2019b, 2023).

7.1 Bridge Card Game

In the domain of formal verification and model checking, games have emerged as valuable frameworks for representing real-world decision-making scenarios. This preference arises from two primary factors: their inherent structural properties that facilitate formal modeling and parametrization, combined with widespread familiarity that enables intuitive explanation of complex concepts. Historically, tabletop games have served as canonical testbeds for verification methodologies, with chess representing a quintessential example of perfect information games where complete state observability (i.e., positions of all pieces) is available to all participants.

However, to investigate systems with information asymmetry, card games offer superior modeling capabilities. These games naturally incorporate imperfect information through concealed player assets - most notably the private hands maintained by participants. This characteristic makes them particularly suitable for studying scenarios where agents must make decisions based on partial observations of the global system state.

This section introduces formal modeling approaches for the well-established card game of bridge, including its absentminded variant, which presents unique challenges for verification frameworks due to its combination of partial observability and memory constraints.

All models presented in this section are synchronous, turn-based, feature asymmetric information distribution among agents, and operate under deterministic temporal progression. These properties ensure that the models accurately reflect the dynamics of bridge gameplay while remaining amenable to formal analysis.

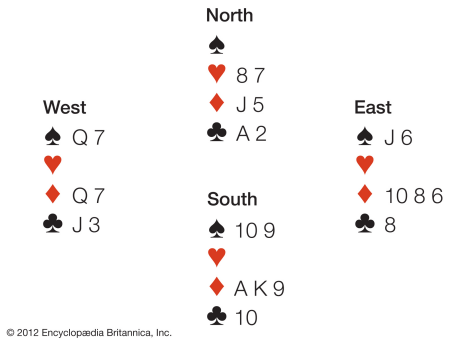

Figure 7.1: Example 6-endplay in bridge

7.1.1 Bridge Model: The Overview

We have analyzed bridge play scenarios that correspond to standard configurations documented in bridge literature and periodicals. The verification objective involves determining an optimal declarer strategy under a NoTrump contract, with the declarer position (South, denoted as **S**) operating within the k-endplay phase of the game (Figure 7.1 provides a schematic representation). The game utilizes a standard deck of $4n$ cards (n cards per suit),[1] where the initial configuration defines each player's hand as containing k cards following the execution of $n - k$ prior plays. This parameterization generates a family of verification models indexed by (n, k) combinations. Victory conditions for the declarer require securing more than $\lceil k/2 \rceil$ tricks during the endplay phase, establishing a quantitative benchmark for strategy evaluation.

The game proceeds in clockwise turn order, with **S** initiating play from the declarer position. Trick resolution follows standard bridge mechanics: each trick consists of four cards played sequentially by participants, with the winner of the current trick leading the subsequent round. The declarer assumes control over both their own hand and the dummy's (**N**) holdings, while opponents (**W** and **E**) manage their respective private hands independently. The dummy's hand remains fully visible to all participants, maintaining asymmetric information distribution where only declarer/defenders maintain private knowledge of their concealed holdings.

Each player's local state representation incorporates five critical components:

[1] While actual bridge employs $n = 13$ suits, our analysis maintains variable parameters to evaluate algorithmic scalability across different model sizes.

- *Current player's hand composition* (private holdings for declarer/dummy or opponents)
- *Dummy's exposed hand contents* accessible to all participants
- *Evolution of played cards* encompassing the complete sequence of discarded cards throughout the game
- *Trick resolution dynamics* tracking card-player associations and leadership transitions within current trick
- *Cumulative scoring metrics* recording trick victories for both declarer and defender factions

This state representation enables formal verification of strategic decision-making under partial observability constraints inherent in bridge gameplay.

Our analysis focuses exclusively on the declarer's strategic capabilities (**S**), rendering the epistemic relations of opposing players (**W**, **E**) inconsequential to the verification objectives. The formal model exhibits four distinctive properties:

- *Turn-based execution with synchronization*: The game progresses through sequential clockwise actions, incorporating explicit *wait* transitions to synchronize player interactions during trick execution phases.
- *Asymmetric information structure*: Strategic uncertainty arises from private holdings - while the dummy's (**N**) hand remains fully observable, opposing hands maintain strict confidentiality. This partial observability directly impacts decision-making efficacy, as corroborated by empirical studies on expert bridge players' strategic limitations[1].
- *Bounded memory representation*: The model implements imperfect recall through abstraction of play sequence details. For instance, after a trick containing ♣10, ♣J, ♣A, and ♣8 in Figure 7.1, the declarer only retains set membership knowledge of played cards rather than their ownership mapping.

[1] This aligns with psychological research indicating that intermediate players typically track played cards but lose specific attribution information, unlike experts who maintain complete play history records

- *Deterministic temporal progression*: The lockstep synchronization mechanism ensures global observability of state transitions, which formally establishes equivalence between tr_{L2} and tr_{L3} team-reasoning relations when evaluating singleton coalitions within the verification framework.

These properties create a rigorous foundation for analyzing strategic reasoning under constraints of partial observability and bounded memory, while the synchronization mechanism enables precise temporal analysis through universally visible state transitions.

7.1.2 Bridge Model: The Details

Our iCGS for the bridge endplay is constructed as follows.

Agents and States

We formalize the game's components as follows. Given model parameters $(n, k) \in \mathbb{N}^2$ where n denotes the number of distinct ranks per suit and k represents the initial hand size, we define:

- **Card Ranks:** $Ranks_n = \{1, \ldots, n\}$ with n corresponding to the Ace (highest rank), $n-1$ to the King, and so forth
- **Card Suits:** $Suits = \{1, \ldots, 4\}$ where suit 1 represents Spades (♠), suit 2 Hearts (♡), suit 3 Diamonds (♢), and suit 4 Clubs (♣)
- **Game Deck:** $Deck_n = Ranks_n \times Suits$
- **Player Positions:** $Pos = \{0, \ldots, 3\}$ with bijection $\mathbf{S} \leftrightarrow 0$, $\mathbf{W} \leftrightarrow 1$, $\mathbf{N} \leftrightarrow 2$, $\mathbf{E} \leftrightarrow 3$
- **Agent Set:** $\mathbb{A}\mathrm{gt} = \{\mathbf{S}, \mathbf{W}, \mathbf{E}\}$, where the declarer ($\mathbf{S}$) controls both South and North positions while $\mathbf{W}$ and $\mathbf{E}$ control their respective hands

The global state space St of the iCGS framework is defined as 8-tuples:

$$(hands, tricks, next, board, lead, history, clock, suit)$$

with the following formal interpretations:

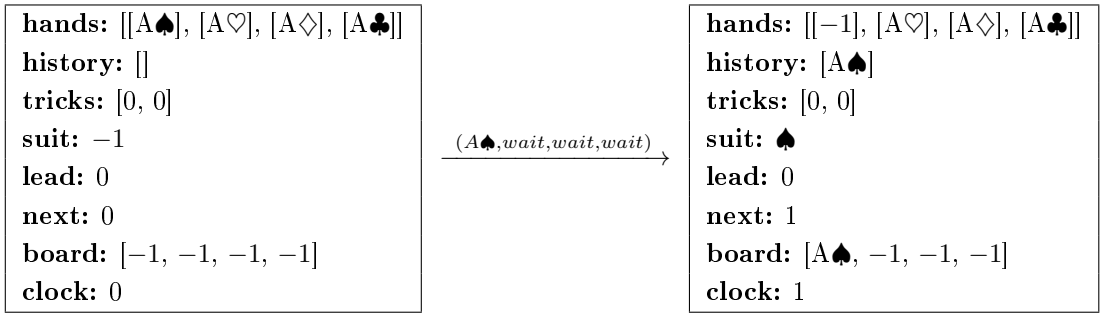

Figure 7.2: Example initial state of a $(1, 1)$ bridge model, with an opening transition

$hands = (h_{\mathbf{S}}, h_{\mathbf{W}}, h_{\mathbf{N}}, h_{\mathbf{E}})$ where each $h_i \subseteq Deck_n$ and $h_i \cap h_j = \emptyset$ for $i \neq j$. Internally represented as four arrays of size k with -1 placeholders for played cards. Card arrays are initially sorted in ascending rank order.

$history \subseteq Deck_n$: Ordered sequence of all cards played during the game, maintained in ascending rank order

$tricks \in \{0, \ldots, k\}^2$: Current trick scores for the declarer team ($tricks_{\mathbf{S}}$) and defender team ($tricks_{\mathbf{W}} + tricks_{\mathbf{E}}$)

$suit \in Suits \cup \{-1\}$: Active lead suit for current round (-1 indicates undefined)

$lead \in Pos$: Position initiating the current round

$next \in Pos$: Position scheduled to play next card

$board = (c_1, \ldots, c_4)$ with each $c_i \in Deck_n \cup \{-1\}$, representing cards currently in play where -1 indicates unplayed positions

$clock \in \{0, \ldots, 4\}$: Round progression phase (0-3: card playing stages; 4: trick resolution stage)

Parts of example bridge models $(1, 1)$ and $(2, 2)$ are shown in Figures 7.2 and 7.3.

Players are numbered from 0 to 3, clockwise, beginning from the declarer. Cards are described as numbers. Following equation is used: $cardNumber = cardValue * 10 + color$, where card value starts at 14 for Ace, 13 for King, ... And color is number from 1 to 4. Example initial state:
{'hands': [[144], [143], [142], [141]], 'tricks': [0, 0], 'next': 0, 'board': [−1, −1, −1, −1],

hands: [[−1, A♡], [−1, A♠], [K♠, −1], [K♣, A♣]]
history: [K♢, K♡, A♢]
tricks: [0, 0]
suit: ♢
lead: 0
next: 3
board: [K♢, K♡, A♢, −1]
clock: 3

Figure 7.3: A state of bridge endplay for $n = 2, k = 2$

'beginning': 0, 'history': [], 'clock': 0, 'suit': −1} Example next state, after declarer played his card:
{'hands': [[−1], [143], [142], [141]], 'tricks': [0, 0], 'next': 1, 'board': [144, −1, −1, −1], 'beginning': 0, 'history': [144], 'clock': 1, 'suit': 4} Another example for better understanding:
{'hands': [[−1, 143], [−1, 144], [134, −1], [131, 141]], 'tricks': [0, 0], 'next': 3, 'board': [132, 133, 142, −1], 'beginning': 0, 'history': [132, 133, 142], 'clock': 3, 'suit': 2}

Actions and Transitions

The action set for each agent $i \in \mathbb{A}\text{gt}$ at state $q \in St$ is defined as:

$$A_i(q) = \begin{cases} \text{AvailableCards}_i(q) & \text{if } i = \text{next}(q) \\ \{\text{wait}\} & \text{otherwise} \end{cases}$$

where $\text{AvailableCards}_i(q)$ corresponds to the set of cards in agent i's hand that conform to the current round's suit requirements. An illustrative transition between states is provided in Figure 7.2.

Epistemic Indistinguishability

Definition 7.1.1 (Declarer's Epistemic Equivalence) *Two states $s, s' \in St$ are epistemically equivalent for the declarer ($\boldsymbol{S}$), denoted $s \sim_{\boldsymbol{S}} s'$, iff:*

1. *All state variables except opponents' hands are identical:*

$$(hands_{\boldsymbol{S}}, tricks, next, board, lead, history, clock, suit)$$

$$=$$

$$(hands'_{\boldsymbol{S}}, tricks', next', board', lead', history', clock', suit')$$

2. The opponents' hands have equal cardinality:

$$|hands_{\boldsymbol{W}}| = |hands'_{\boldsymbol{W}}| \quad and \quad |hands_{\boldsymbol{E}}| = |hands'_{\boldsymbol{E}}|$$

This equivalence captures the declarer's limited knowledge about opponents' specific card distributions. For instance, the state depicted in Figure 7.3 maintains equivalence under $\sim_{\mathbf{S}}$ when swapping the Ace of Hearts ($A\heartsuit$) and King of Clubs ($K\clubsuit$) between opponents, resulting in:

$$hands = ([-1, A\heartsuit], [K\clubsuit, -1], [K\spadesuit, -1], [A\clubsuit, A\spadesuit])$$

where all declarer-visible state components remain unchanged.

7.1.3 Bridge Endplay by Absentminded Declarer

In bridge endplay models, agents maintain perfect observability of action execution throughout gameplay. Consequently, for singleton coalitions, the steadfast next-time operator $\langle a \rangle^{\bullet}$ becomes logically equivalent to the standard next-time modality $\langle a \rangle$. To evaluate the robustness of our lower bound construction from Section 3.1, we introduce a variant scenario where the declarer exhibits partial observability: the agent remains unaware of card placements during trick execution and can only perceive the final outcome. Furthermore, this modified framework permits the declarer to schedule card plays from both her own and the dummy's hand concurrently with opponent actions. Such architectural modifications lead to expanded indistinguishability relations for $\mathbf{S}$ while inducing significant growth in both state space cardinality and transition graph complexity.

These adjustments affect the available actions for the declarer, as she can now play her cards without awaiting opponent moves. The epistemic equivalence relation $\sim_{\mathbf{S}}$ is redefined to account for the declarer's limited knowledge regarding card placements during tricks. Specifically, two states are considered indistinguishable if they differ solely in the specific cards held by opponents, provided that the number of cards in each opponent's hand remains consistent across both states. This alteration reflects the declarer's inability to track individual card distributions among defenders during trick execution, thereby encapsulating the intended partial observability within the model.

The revised model retains the core properties of turn-based execution, asymmetric information distribution, bounded memory representation, and deterministic temporal

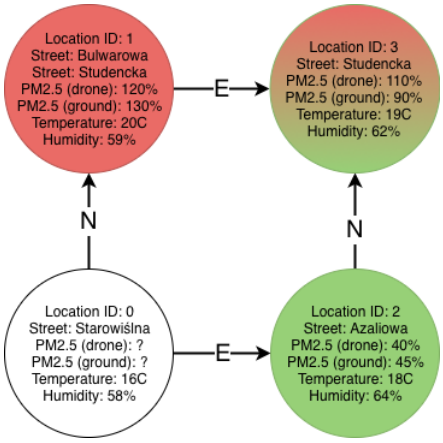

Figure 7.4: Map: drone navigation and measurements in an area of Cracow. Location colors indicate whether the PM2.5 readings are within or beyond the norm

progression. However, the synchronization mechanism is adapted to accommodate the declarer's concurrent action scheduling, ensuring that state transitions remain globally observable despite the introduced partial observability constraints. This framework facilitates rigorous analysis of strategic reasoning under modified informational conditions while preserving the foundational structure necessary for formal verification.

7.2 Drone Teams

A coordinated team of autonomous drones represents an ideal application scenario for multi-agent systems. The architectural framework of this system comprises multiple autonomous agents, specifically unmanned aerial vehicles (UAVs). An illustrative implementation of this paradigm is subsequently described.

Models presented in this section are considered to be synchronous.

7.2.1 Drones Patrolling for Pollution

Consider a coordinated fleet of k autonomous unmanned aerial vehicles (UAVs) deployed for air quality monitoring in Kraków, Poland. The operational environment employs a simplified urban grid representation (see Figure 7.4), comprising four discrete monitoring locations interconnected through unidirectional movement pathways constrained to northward and eastward trajectories. All agents originate from location 0 and are tasked with reaching location 3, designated as the terminal target location.

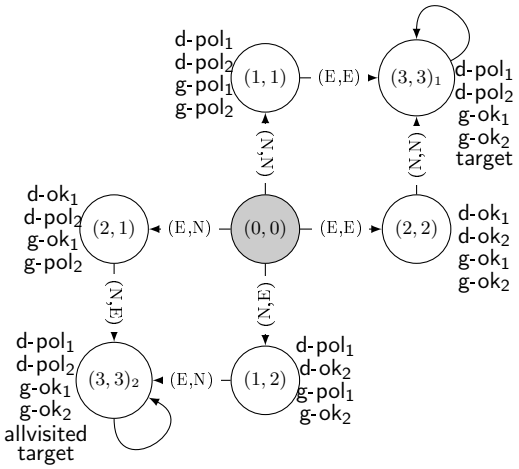

Figure 7.5: Model M_1: autonomous drones monitoring pollution

Each UAV integrates onboard sensing capabilities to quantify atmospheric particulate matter (PM2.5) concentrations, defined as airborne particles with diameters less than 2.5 microns. Additionally, UAVs establish communication with proximate ground-based environmental stations to obtain synchronized measurements of surface-level PM2.5, temperature, barometric pressure, and humidity. Notably, baseline measurements are unavailable at the initial location (0). For analytical tractability, the environmental parameters are assumed static throughout the mission duration, eliminating temporal variability in pollutant dispersion patterns and meteorological conditions.

Fig. 7.5 illustrates a formal Concurrent Game Structure (CGS) model $\mathcal{M}_{\text{drones}}$ representing the described operational scenario with $k = 2$ UAVs. Each UAV operates as a distinct autonomous agent with two primitive navigation actions: N (northward traversal) and E (eastward traversal). The system incorporates operational constraints through bounded mobility limits ($l = 2$) simulating finite battery capacity. The state space encodes both current positional data and visited location history, with positional coordinates represented as tuples (x_1, x_2) where x_i denotes UAV i's location. Route-specific state differentiation is implemented only for $(3, 3)_1$ and $(3, 3)_2$, corresponding to the two unique path combinations enabling co-terminal occupancy at the target location. This state distinction mechanism preserves path-dependent environmental observation patterns while maintaining computational tractability.

We define the following atomic propositions for characterizing environmental monitoring scenarios:

- $\mathsf{d\text{-}pol_i}$: Indicates drone i's sensor detects elevated PM2.5 levels exceeding regulatory thresholds.
- $\mathsf{d\text{-}ok_i}$: Signifies drone i's sensor measures PM2.5 concentrations within acceptable limits.
- $\mathsf{g\text{-}pol_i}$: Denotes the nearest ground-based sensor to drone i registering PM2.5 levels above norms.
- $\mathsf{g\text{-}ok_i}$: Corresponds to the nearest ground sensor measuring PM2.5 within permissible ranges.
- target: Labels states where all drones in the system have successfully reached the designated target location.
- $\mathsf{allvisited}$: Represents complete coverage of all spatial locations in the operational map. For the specific model M_1, this proposition holds only in state $(3,3)_2$, which constitutes the terminal coverage condition.

These propositions form the basis for formal verification of coordinated drone navigation protocols in urban air quality monitoring applications.

For the presented model the following example formula holds: $M_1, (0,0) \models \langle\langle 1 \rangle\rangle \mathbf{F}\mathsf{d\text{-}pol_1}$: drone 1 has a strategy ensuring that its sensor will eventually register pollution. The strategy itself is simple: when in the state $(0,0)$ fly North. Moreover, $M_1, (0,0) \models \langle\langle 1 \rangle\rangle \mathbf{F}\mathsf{d\text{-}ok_1}$: it also has a strategy for reaching a location where it registers no pollution. This time the simplest strategy is to fly East. In fact, $M_1, (0,0) \models \langle\langle 1 \rangle\rangle (\mathbf{F}\mathsf{d\text{-}pol_1} \wedge \mathbf{F}\mathsf{d\text{-}ok_1})$: there is a single strategy for achieving both goals. The drone needs to fly East first, and then fly North.

Furthermore, no drone can assure on its own that all of the locations will eventually be visited: $M_1, (0,0) \models \neg\langle\langle 1 \rangle\rangle \mathbf{F}\mathsf{allvisited} \wedge \neg\langle\langle 2 \rangle\rangle \mathbf{F}\mathsf{allvisited}$. This can only be ensured if both drones cooperate: $M_1, (0,0) \models \langle\langle 1,2 \rangle\rangle \mathbf{F}\mathsf{allvisited}$. On the other hand, the drones are bound to end up at the target location, no matter what they decide to do: $M_1, (0,0) \models \langle\langle \emptyset \rangle\rangle \mathbf{F}\mathsf{target}$.

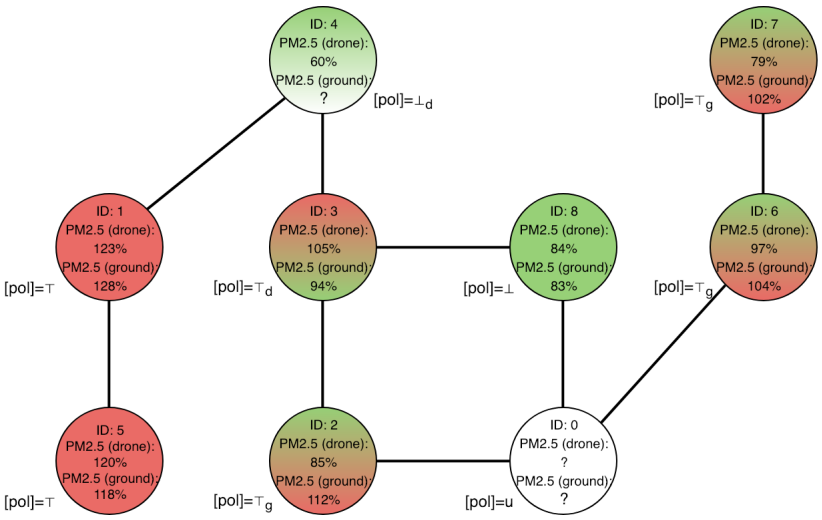

Figure 7.6: The map used in the experiments

7.2.2 Drones Benchmark

The benchmark architecture represents a formal extension of the drone swarm model developed in prior sections. We conceptualize the operational environment as a directed graph $G = (Loc, E)$, where Loc denotes the discrete set of spatial coordinates defining permissible drone positions, and $E \subseteq Loc \times Loc$ encodes the feasible navigation pathways between these locations. As depicted in Figure 7.6, the experimental configuration assumes bidirectional connectivity for all edges $e \in E$, thereby enabling representation through an undirected graph structure without loss of navigational fidelity. This abstraction allows for streamlined analysis of multi-agent coordination dynamics while maintaining the topological constraints inherent to urban airspace navigation.

The system architecture comprises a finite set of autonomous drone agents $D = \{d_1, d_2, ..., d_n\}$ operating within an environmental framework. Each drone d_i is equipped with onboard sensors to quantify localized pollution concentrations $p_i(t) \in \mathbb{R}_{\geq 0}$ at its current spatial position $l_i(t) \in Loc$, where Loc represents the discrete location set defined in the map graph $G = (Loc, E)$. Communication capabilities are implemented through Bluetooth Low Energy (BLE) protocols, enabling bidirectional exchange of sensor data between agents co-located or within adjacent vertices in G, forming a dynamic local adjacency network.

Global environmental data from terrestrial sensor arrays are disseminated via a centralized monitoring infrastructure, ensuring universal accessibility to all agents throughout operational timelines. This distributed sensing paradigm is formalized through an epistemic indistinguishability relation $\sim_i$ for each drone d_i, where equivalence classes correspond to indistinguishable system states based on the following observables:

- Current spatial coordinates $l_i(t) \in \mathit{Loc}$;
- Local sensor measurement $p_i(t)$;
- Networked sensor data $\{p_j(t)\}_{j \in D, j \sim_i}$;
- Global terrestrial sensor array readings R_{ground};
- Real-time battery state metrics $b_i(t) \in \mathbb{R}_{\geq 0}$;
- Historical trajectory data $\{l_i(\tau)\}_{\tau < t}$ of previously visited locations.

This epistemic framework establishes the informational basis for decentralized decision-making in heterogeneous multi-agent systems.

We establish a temporal constraint of $T_{\max} = 30\,\text{min}$ for the operational mission duration, reflecting current technological limitations in consumer-grade unmanned aerial vehicles (UAVs) where sustained flight beyond this threshold remains restricted to specialized industrial and military platforms. This constraint enables the formulation of a quasi-static environmental model, wherein pollution concentrations maintain spatial invariance across the mission timeline. Formally, the pollution profile $P : \mathit{Loc} \to \mathbb{R}_{\geq 0}$ maps each location $l \in \mathit{Loc}$ to a fixed contamination level $P(l)$, ensuring deterministic sensor readings for repeated visits to identical spatial coordinates. This assumption simplifies environmental state tracking while preserving critical dynamics for multi-agent coordination optimization.

Each autonomous agent $d_i \in D$ executes actions from the set $\mathcal{A} = \{\text{North}, \text{South}, \text{East}, \text{West}, \text{Wait}\}$, where directional actions correspond to edge traversals in the graph G and Wait preserves the current vertex. Energy expenditure ϵ_a is action-dependent, with movement actions consuming $\epsilon_m > 0$ units per traversal, while Wait incurs no energy cost. The battery state $b_i(t)$ evolves according to:

$$b_i(t+1) = \max\left(0,\, b_i(t) - \epsilon_a \cdot \mathbf{1}_{\text{movement}}(a_i(t))\right)$$

When $b_i(t) = 0$, the agent's action space reduces to $\mathcal{A}_i^{\text{crit}} = \{\text{Wait}\}$, enforcing perpetual stasis at location $l_i(t)$ due to the absence of recharging mechanisms in our model. Critically, immobilized agents retain communication functionality, enabling persistent broadcasting of sensor data to adjacent operational drones within the BLE network topology.

The system dynamics are parameterized by:

- Population cardinality $N = |D|$, governing swarm size;
- Initial battery capacity $B_0 \in \mathbb{R}_{>0}$, uniformly assigned across agents.

These parameters define the operational constraints for energy-aware path planning in persistent environmental monitoring tasks.

7.3 Machines and Robots

We present a formal framework for resource coordination in Industry 4.0 environments, focusing on smart production factories (SPFs) Schlingloff (2018). In these systems, heterogeneous machinery distributed across geographically distinct factory nodes forms interconnected production chains where intermediate components serve as prerequisites for downstream manufacturing processes. This spatially distributed architecture necessitates precise orchestration of material flows between functionally specialized equipment units, creating complex dependencies in both temporal and logistical dimensions. Our analysis addresses the technical challenges inherent in maintaining production continuity while optimizing energy efficiency and throughput in such decentralized manufacturing ecosystems.

Traditionally, material transportation in industrial environments is accomplished using automated guided vehicles (AGVs) with fixed routing systems. In contrast, the Smart Production Framework (SPF) employs a fleet of autonomous transport robots (ATRs), as illustrated in Fig. 7.7, capable of dynamic path planning and unrestricted navigation through structured factory environments. These ATRs operate independently of physical guidance infrastructure, relying instead on advanced localization algorithms. The robotic fleet typically comprises 4–20 units, with individual payload capacities ranging from 50–200 kg. Each ATR utilizes a predefined high-fidelity map of the facility (Fig. 7.9), which encodes spatial constraints including prohibited zones,

Figure 7.7: Loading of a transport robot.

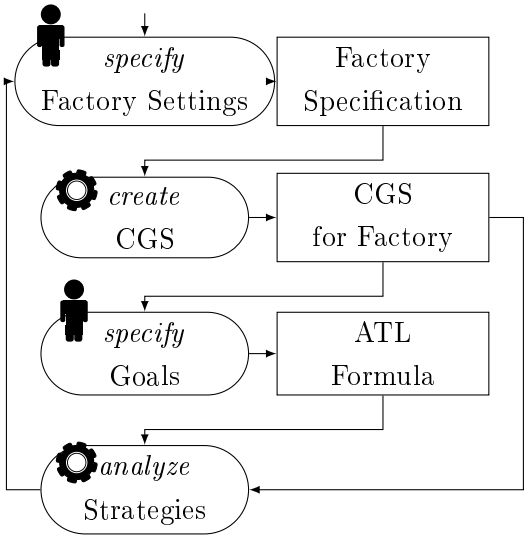

Figure 7.8: Verification approach showing manual (stick-figure) and automated (gear-wheel) activities (rounded rectangles) and corresponding artifacts (rectangles).

charging station locations, and critical infrastructure positions. Real-time environmental perception is achieved through LiDAR sensor arrays, which generate precise three-dimensional spatial data. This sensor data undergoes continuous fusion with the reference map through simultaneous localization and mapping (SLAM) algorithms to maintain positional accuracy. Power management is facilitated by lithium-ion battery systems requiring periodic recharging through automated docking procedures. The system implements machine-to-robot (M2R) and robot-to-robot (R2R) communication protocols via industrial wireless networks, enabling real-time coordination of material handling tasks based on production demands.

Compared to traditional automated guided vehicles, autonomous transport robots

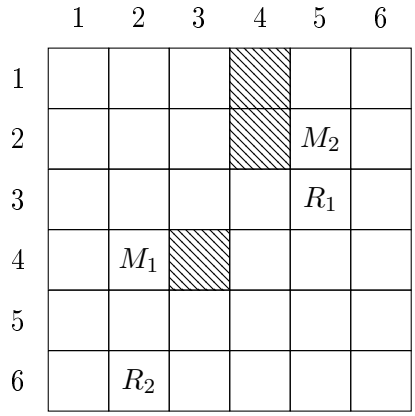

Figure 7.9: Conceptual representation of factory map with three obstacles (dashed parcels), two machines M_1 and M_2, and two robots R_1 and R_2.

provide the following benefits:

- Scalable deployment: ATR fleets can be expanded through dynamic integration of additional units during active factory operations without requiring infrastructure modifications or operational disruption.
- Agile spatial adaptation: Reconfiguration of machine locations only necessitates updating target coordinates in the reference map, eliminating the need for route reprogramming while maintaining navigation efficiency.
- Fault-tolerant operation: Mechanical failure of individual ATRs does not disrupt production workflows due to autonomous task redistribution among functional units via real-time path optimization.
- Dynamic obstacle negotiation: ATRs employ sensor fusion and reactive path planning algorithms to navigate around environmental obstructions or personnel without human intervention.
- Predictive workload balancing: Idle ATRs interface with high-productivity workcells through autonomous reallocation, utilizing predictive workload analytics to optimize resource utilization across the production network.

To enable systematic analysis of context-dependent production requirements, we implemented a formal methodology combining domain-specific modeling and strategic

verification. As depicted in the figure, our approach begins with the development of a domain-specific language (DSL) for characterizing production environments, accompanied by an automated toolchain for translating DSL specifications into formal CGS models. Through this framework, we formalized multiple scenario variations and automatically synthesized corresponding CGS models. Operational objectives were then encoded as ATL formulae over the CGS structures, enabling formal verification of cooperative strategies. These specifications underwent rigorous evaluation through strategic model checking techniques implemented using the algorithmic framework detailed in Chapter 3.

7.3.1 Modelling Smart Production Factories

Factory Layout The factory layout specification formally defines three critical spatial dimensions: (1) the facility's geometric extent represented as a matrix of discrete floor tiles, (2) static structural elements including obstacles and machinery, and (3) the initial deployment configuration of operational autonomous transport units.

Example 7.3.1 (Canonical Factory Configuration) *Consider the* 6×6 *facility matrix illustrated in Fig. 7.9, which establishes a 36-tile working environment. This configuration designates three obstruction zones located at coordinates* $(3,4)$*,* $(4,1)$*, and* $(4,2)$*. Two production nodes are deployed at* $(2,4)$ *(designated* M_1*) and* $(5,2)$ *(*M_2*) respectively. The robotic deployment consists of two operational units initialized at* $(2,6)$ *(*R_2*) and* $(5,3)$ *(*R_1*), with their positions recorded in the system's reference coordinate frame.*

Machine Production Requirements The formal specification of production systems requires precise characterization of machine-specific item dependencies. Our framework introduces a formal syntax to define transformation ratios where item types correspond to machine identifiers within the production hierarchy.

Example 7.3.2 (Production Dependency Specification) *For machine* M_2 *in the configuration from Ex. 7.3.1, consider the dependency mapping:*

$$M_1 \mapsto 1 \cdot M_2$$

This formal expression defines that machine M_1 *requires one unit of output from* M_2 *to execute its production process. Such mappings establish the foundational relationships in the facility's material flow network.*

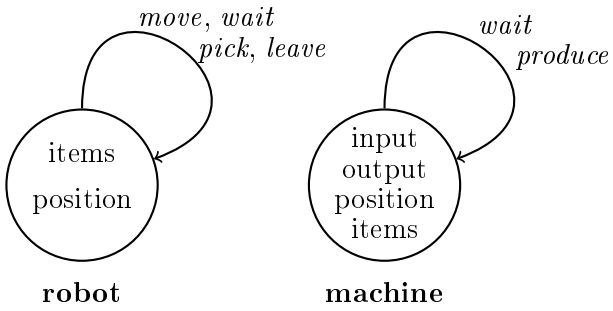

Figure 7.10: CGS for robots and machines.

The language further incorporates production capacity constraints through explicit upper bounds on item generation rates. This constraint serves to bound the state-space complexity of the contextual goal structure (CGS) by avoiding combinatorial explosion inherent in unbounded production systems. Each machine specification includes a maximum throughput parameter $\mu \in \mathbb{N}$ that limits its contribution to the global state transition graph.

From the Specification to the Model This formal specification serves as the foundation for systematic analysis through automated synthesis of a contextual goal structure (CGS). The CGS model incorporates two distinct agent categories: *autonomous transport robots* (ATRs) responsible for material handling, and *production nodes* representing manufacturing entities. As illustrated in Fig. 7.10, each agent type maintains a hierarchical state representation coupled with domain-specific action spaces. The ATRs' states encode spatial coordinates, battery levels, and cargo status, while production nodes track input/output buffers and processing cycles. This hierarchical architecture enables formal verification of cooperative strategies while preserving spatial constraints and operational dependencies inherent in the factory layout. The figure's annotated transitions demonstrate the multi-agent coordination protocol governing material flow and resource allocation.

Robot Agent Specification The autonomous transport robot (ATR) agent specification defines a state vector $\mathbf{S}_R = (x, y, c)$ comprising three fundamental parameters:

- *Positional coordinates*: A 2D Cartesian tuple $(x, y) \in \mathbb{N}^2$ representing valid factory floor tile locations

- *Cargo state*: An integer $c \in \mathbb{N} \cup \{-1\}$ where $c = -1$ indicates empty cargo hold and $c = m$ (with m being a machine identifier) denotes carriage of item type m

The robot's action space $\mathbb{A}_R$ includes:

- *Locomotion commands*: $\{\text{NORTH}, \text{SOUTH}, \text{EAST}, \text{WEST}, \text{STATIONARY}\}$ with movement validity determined by factory boundaries and obstacle positions. Collision detection occurs when multiple robots occupy identical coordinates (x, y), resulting in persistent immobilization of involved units.
- *Material handling operations*: Conditional execution of $\{\text{PICK}, \text{DELIVER}\}$ actions requires spatial co-location with production nodes. These actions modify the cargo state c through bidirectional item transitions between robotic agents and manufacturing entities.

Production Node Specification The manufacturing entity (ME) agent specification formalizes a state vector $\mathbf{S}_M = (x, y, \mathbf{I}, O, T)$ containing:

- *Geospatial coordinates*: Machine location $(x, y) \in \mathbb{N}^2$ within the facility's tile matrix
- *Input buffer vector*: $\mathbf{I} \in \mathbb{N}^n$ where n equals the number of distinct machine identifiers, with I_m representing the current count of type-m items required for production
- *Output buffer scalar*: $O \in \mathbb{N}$ indicating the number of completed items awaiting robotic retrieval
- *Production counter*: $T \in \mathbb{N}$ tracking total items manufactured, crucial for bounding the CGS state-space complexity

The ME action space $\mathbb{A}_M = \{\text{WAIT}, \text{PRODUCE}\}$ operates under strict preconditions:

- `WAIT` maintains current state until all input buffers satisfy production requirements $\forall m \in \mathcal{M},\ I_m \geq \rho_m$ where ρ_m is the specified input ratio
- `PRODUCE` triggers state transition $(x, y, \mathbf{I}, O, T) \to (x, y, \mathbf{I} - \boldsymbol{\rho}, O + 1, T + 1)$ when input thresholds $\boldsymbol{\rho}$ are met, decrementing input buffers and incrementing both output and production counters

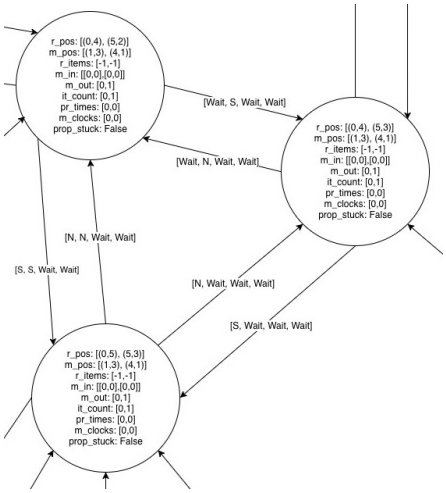

Figure 7.11: Fragment of the CGS produced for the example factory setting specified in Ex. 7.3.1 and Ex. 7.3.2.

Example 7.3.3 (CGS for simple factory) *The CGS representation for the canonical factory configuration from Ex. 7.3.1 and Ex. 7.3.2 contains* 3,581 *reachable states. As shown in Fig. 7.11, the formal model encodes multi-agent coordination through structured state variables:*

$r\_pos \in \mathbb{N}^2$	*Robot positional coordinates*
$m\_pos \in \mathbb{N}^2$	*Machine positional coordinates*
$r\_items \in \mathbb{N} \cup \{-1\}$	*Robot cargo status*
$m\_in \in \mathbb{N}^n$	*Machine input buffer vector*
$m\_out \in \mathbb{N}$	*Machine output buffer count*
$it\_count \in \mathbb{N}$	*Cumulative production counter*
$pr\_times \in \mathbb{N}^n$	*Production cycle duration vector*
$m\_clocks \in \mathbb{N}^n$	*Production progress timers*
$prop\_stuck \in \{0,1\}^n$	*Buffer overflow indicators*

The terminal state fragment in Fig. 7.11 represents two production nodes at positions $(1,3)$ *and* $(4,1)$*, both with empty input buffers. The second machine maintains* $m\_out = 1$ *pending item awaiting retrieval. Two ATRs occupy identical coordinates with* $r\_items = -1$*, indicating unladen status. The state transition analysis demonstrates that robot* R_1 *can initiate a NORTH movement command to transition to the adjacent state node in the upper-right quadrant, validating the formalization of locomotion dynamics within the CGS framework.*

7.3.2 Model Extensions for Operational Realism

To enable comprehensive analysis of industrial automation scenarios, we implemented three incremental model extensions that enhance the basic framework's fidelity while maintaining formal verification capabilities:

- **Energy-aware Mobility:** The extended model introduces battery state variables $B_r \in \mathbb{N}$ for each robot r, alongside charging station locations $C_s \in \mathbb{N}^2$. Movement actions decrement B_r by one unit per tile traversal, while charging operations restore full capacity $B_{\max}$ through dedicated docking states. This enables energy-constrained path planning and strategic analysis of charging station utilization.

- **Transient Storage Dynamics:** Intermediate storage zones $S_k = (x_k, y_k, \gamma_k)$ with capacity $\gamma_k \in \mathbb{N}$ are incorporated as material staging areas. Robots can deposit $c = m$ items at co-located storage units, effectively decoupling production node output buffers from transportation delays. This extension facilitates buffer overflow mitigation strategies and enables multi-hop logistics planning.

- **Temporal Production Constraints:** Machine-specific production durations $\tau_m \in \mathbb{N}$ are formalized through progress timers $t_m \in \{0, \ldots, \tau_m\}$. Manufacturing transitions require τ_m consecutive processing steps, with t_m tracking elapsed time since production initiation. This temporal formalism necessitates time-aware scheduling algorithms and introduces deadline-sensitive coordination requirements.

These extensions systematically increase the model's operational realism while preserving its formal verification properties. The energy-constrained model enables analysis of battery management strategies, storage-aware configurations support buffer management optimization, and time-dependent production formalizes temporal coordination constraints in manufacturing workflows. Each modification introduces new state variables and transition conditions that expand the CGS's representational capacity while maintaining computational tractability through structured state-space partitioning.

7.4 Social Explainable AI

In today's digital age, artificial intelligence (AI) seamlessly weaves itself into our daily routines, guiding us from social media exchanges to vehicle routing, and even shaping our entertainment choices. But its influence extends beyond individual convenience. The corporate sphere heavily leans on AI, marking pivotal changes in societal and economic landscapes.

The current momentum in AI research is gravitating towards a paradigm termed as *Social Explainable AI (SAI)*, an emerging movement that emphasizes decentralization, a human-focused approach, and clarity Contucci et al. (2022); Social Explainable AI, CHIST-ERA (2021–24). This trend is both a reaction to the mounting skepticism about traditional, centralized machine learning models and a recognition of challenges like scalability. There's also a deeper ethical dimension, emphasizing the importance of transparent data practices and the reliability of computational systems Drainakis et al. (2020); Ottun et al. (2022).

Though SAI is a nascent idea, it holds a rich potential for research. As researchers delve into this area, it's paramount to evaluate whether SAI truly lives up to its promises of effectiveness, clarity, and user-centricity. A comprehensive grasp of SAI entails scrutinizing its intended features and probing its unforeseen dynamics, especially when AI tools and humans intertwine Conti and Passarella (2018); Fuchs et al. (2022); Toprak et al. (2021). There's an acute apprehension about SAI's vulnerability to malicious threats like data impersonation or intricate cyber-attacks. If SAI lacks robustness against these threats, its weak points could become prime targets.

While adversarial threats to traditional machine learning aren't a new topic, the discourse around SAI has largely been preoccupied with its prospective functionalities. This could be due to the intricate challenges—be they conceptual, computational, or societal—that SAI brings along. Probing into the unanticipated behaviors of these systems certainly brings a host of complexities to the forefront.

In this section, we initiate our discussion with a comprehensive description of the SAI framework. Following this, we elucidate our approach to modeling the PAIV network within the S framework, diving into potential threats embedded within the system. Models are subsequently formalized using asynchronous semantics, where global model is derived from local components.

7.4.1 Framework

The framework of SAI Contucci et al. (2022); Fuchs et al. (2022); Social Explainable AI, CHIST-ERA (2021–24) aims to address several drawbacks inherent to the currently dominant AI paradigm. Contemporary ML-based AI systems, by virtue of their scale and the intricacy of their neural network structures, often operate as enigmatic black boxes, eluding even expert understanding. This raises significant *privacy* and *trust* concerns. The centralized storage of ever-growing sensitive data is becoming not just technically challenging but also problematic due to varying data ownership regulations.

Contrastingly, SAI offers a refreshing perspective on ML-based AI systems, focusing on:

- **Individuation**: proposing a "Personal AI Valet" (PAIV) for every individual, serving as their representative in a vast network of interconnected PAIVs;
- **Personalization**: PAIVs employing AI models that are not just explainable but are also customized for individuals;
- **Purposeful interaction**: PAIVs collaboratively building global AI models or making collective decisions based on localized models;
- **Human-centricity**: AI algorithms and PAIV dynamics are anchored in measurable models of individual and societal human behaviors;
- **Explainability by design**: Advancing ML techniques by integrating quantifiable human behavioral models and network science assessments.

Modern SAI developments harness *gossip learning* as the primary ML framework for PAIVs Hegedüs et al. (2019, 2021); Social AI gossiping. Micro-project in Humane-AI-Net (2022). While tools like Lorenzo et al. (2022) offer experimental simulations of this process, our research takes a unique trajectory, concentrating on multiagent interactions during the learning process. We model the PAIV network as an *asynchronous multiagent system (S)*, and formalize its properties as formulas of *alternating-time temporal logic (***ATL****)*.

The foundational step in verifying agent interactions within Social Explainable AI is an in-depth analysis of the underlying protocols. This involves understanding the actions and communications by the systems engaged in the learning process. From

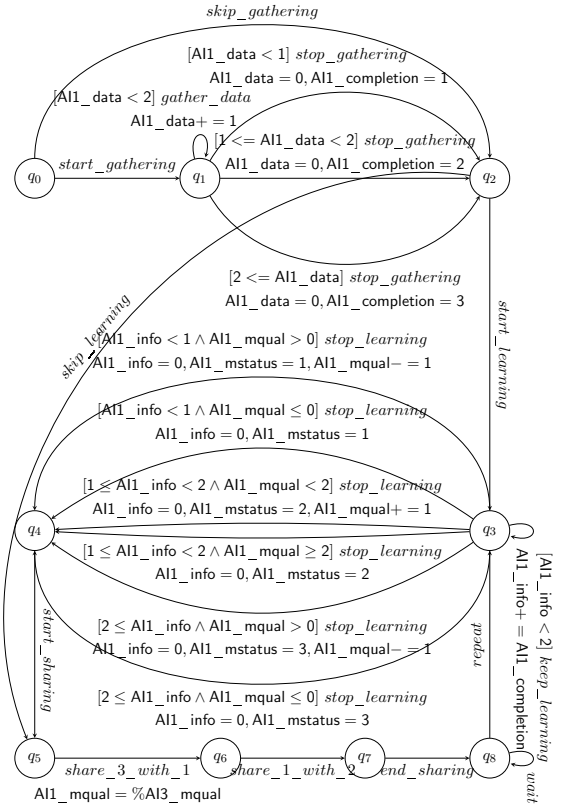

Figure 7.12: Honest AI agent AMAS

there, we can transition to designing multiagent models. Typically, these systems' complexity necessitates abstraction, leading us to craft a more generalized view of the system.

7.4.2 Models

In the context of this research, our primary attention is devoted to the learning facet of the SAI protocol. We visualize each computational unit furnished with an AI module as an individual agent. The localized operational model of an AI agent is delineated into a tri-phase structure: data acquisition, model training, and model dissemination.

Data Acquisition Phase During this initial stage, the agent has the capability to amass the essential data that would feed into the subsequent learning process. This data collection action can be executed repeatedly, and with every iteration, there's

```
start_gathering_data: start -> gather
gather_data: gather -[AI1_data<2]> gather
        [AI1_data+=1]
%% 1: Incomplete data
stop_gathering_data: gather -[AI1_data < 1]> data_ready
        [AI1_data=0, AI1_data_completion=1]
%% 2: Complete data
stop_gathering_data: gather -[1 <= AI1_data < 2]> data_ready
        [AI1_data=0, AI1_data_completion=2]
%% 3: Too much data
stop_gathering_data: gather -[2 <= AI1_data]> data_ready
        [AI1_data=0, AI1_data_completion=3]
skip_gathering_data: start -> data_ready
```

Figure 7.13: Honest AI agent specification in STV - data acquisition phase

a cumulative increment in the local variable signifying the volume of accrued data. Upon culmination of this phase, the amassed data undergoes scrutiny. Based on the resultant value, the agent's preparatory status can be categorized as either incomplete, complete, or exceeding the necessary threshold. With this assessment in place, the agent seamlessly transitions to the model training phase.

Model Training Phase At this juncture, the agent leverages the data stockpiled during the preceding phase to cultivate its local AI model. The training's efficacy is intrinsically tied to the volume of data at the agent's disposal. An abundance of data can predispose the model to overfitting, whereas a data deficit might necessitate multiple training iterations to adequately hone the model. The training procedure can be reiterated several times, subsequently amplifying the local variable that gauges the model's caliber. As this phase wraps up, the model's status is assessed as overfit, underfit, or optimally trained. Subsequent to this evaluation, it becomes imperative for the agent to share its model amongst its peers.

Model Sharing Phase During this stage, agents focus on a mutual exchange of their local AI models, governed by a rudimentary sharing protocol. This protocol finds its foundation in the packet progression mechanics of a ring topology. Each agent inherits the model from its predecessor (in terms of ID) and bequeaths its model to its successor. To bring this cycle full circle, the terminal agent in the sequence shares its model with

```
start_learning: data_ready -> learn
keep_learning: learn -[AI1_information < 2]> learn
        [AI1_information+=AI1_data_completion]
%% 1: Incomplete model
stop_learning: learn
        -[AI1_information < 1 and AI1_model_quality > 0]> educated
        [AI1_information=0, AI1_model_status=1, AI1_model_quality-=1]
stop_learning: learn
        -[AI1_information < 1 and AI1_model_quality <= 0]> educated
        [AI1_information=0, AI1_model_status=1]
%% 2: Complete model
stop_learning: learn
        -[1 <= AI1_information < 2 and AI1_model_quality < 2]> educated
        [AI1_information=0, AI1_model_status=2, AI1_model_quality+=1]
stop_learning: learn
        -[1 <= AI1_information < 2 and AI1_model_quality >= 2]> educated
        [AI1_information=0, AI1_model_status=2]
%% 3: Overtrained model
stop_learning: learn
        -[2 <= AI1_information and AI1_model_quality > 0]> educated
        [AI1_information=0, AI1_model_status=3, AI1_model_quality-=1]
stop_learning: learn
        -[2 <= AI1_information and AI1_model_quality <= 0]> educated
        [AI1_information=0, AI1_model_status=3]
skip_learning: data_ready -> sharing
```

Figure 7.14: Honest AI agent specification in STV - model training phase

the inaugural agent. To stave off potential gridlocks in this process, agents with odd IDs are programmed to first assimilate a model and then distribute theirs. In contrast, agents bearing even IDs prioritize dispatching their model followed by the reception of their predecessor's model.

Upon receiving a model, agents are vested with the discretion to either embrace or rebuff it. This choice is fundamentally influenced by the perceived quality of the incoming model. On endorsing a model, the agent integrates it with its native model, with the resultant model's quality being the superior of the two.

Post the sharing phase, agents have the autonomy to revert to the model training phase, should they wish to refine their models further.

For a more graphic representation, Figure 7.12 presents a local model of an honest AI component. Figures 7.13, 7.14 and 7.15 show the Strategic Verifier (STV) code that delineates its behavior. For a more immersive visualization of this component, one can refer to Figure 7.16, which has been rendered using our tool. STV is our state-of-

```
start_sharing: educated -> sharing
%% Share local model and get average quality of both models
%% receive left
shared share_3_with_1: sharing -> sharing2
        [AI1_model_quality=%AI3_model_quality]
%% send right
shared share_1_with_2: sharing2 -> sharing3
end_sharing: sharing3 -> end
```

Figure 7.15: Honest AI agent specification in STV - model sharing phase

the-art model checker for strategic abilities under imperfect information, which will be discussed in detail in Chapter 8. It uses an agent-based modeling approach, allowing for a modular and intuitive representation of multi-agent systems. Models in STV are constructed using templates that define the behavior of individual agents, which can then be instantiated and composed to form a complete system. More details about the modeling approach of STV can be found in Section 8.1.2.

The global model of a system that consists of a sole honest agent is visualized in Figure 7.17. Similarly, a model for two honest agents is shown in Figure 7.18.

The model that integrates two honest agents, as displayed in Figure 7.18, exemplifies the well-known state-space explosion problem. As a result of this exponential growth in possible states due to the interactions of the two agents, the model becomes highly intricate. This complexity renders the model challenging to interpret and comprehend manually, underscoring the inherent difficulties in managing such dense systems.

7.4.3 Potential Threats in the SAI Ecosystem

The model described in Section 7.4.2 serves as a representation of an optimal circumstance wherein each agent operates with integrity and abides by the prescribed protocol. However, such an idealistic scenario is not always encountered in practice. Various contingencies can arise, such as machines experiencing malfunctions, leading them to undertake actions not sanctioned by the protocol. Moreover, an agent may fall victim to malevolent software, causing it to deviate from its standard functioning. Such aberrations open the door for two potential adversarial scenarios: the *man-in-the-middle* attack and the *impersonator* attack.

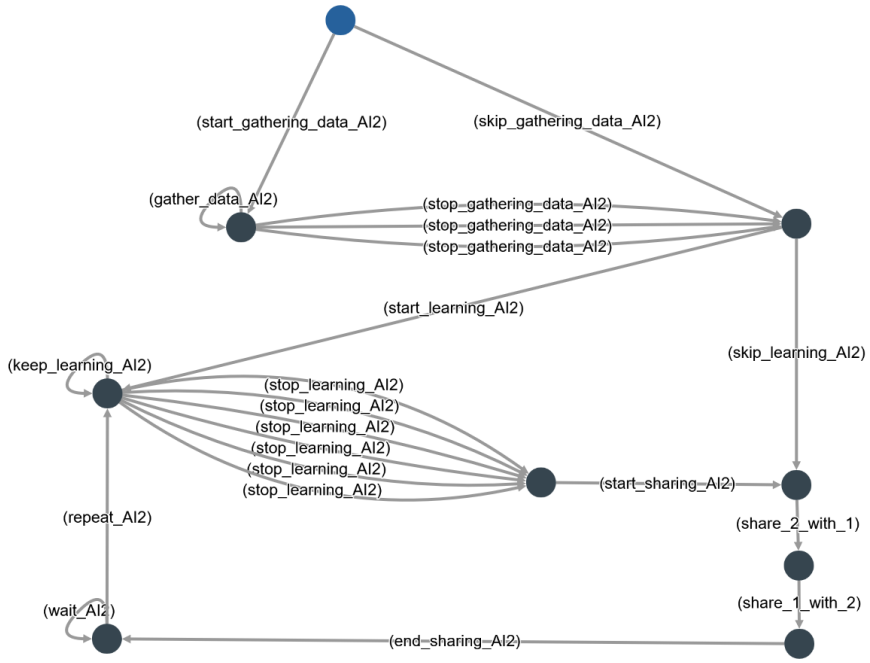

Figure 7.16: Visualization of honest AI in STV

Man-in-the-middle attack Imagine a situation where there's an underhanded agent, termed the *intruder*, lurking within the network. While this nefarious entity remains conspicuously absent during the data acquisition and model training phases, its primary focus and malevolent intent are directed towards the model sharing phase. This intruder possesses the capability to eavesdrop on and intercept any AI model being relayed by a bona fide agent, subsequently forwarding it to another agent within the system. A comprehensive breakdown of the tactics employed by a man-in-the-middle attacker can be found in the STV code depicted in Figure 7.19. Additionally, a visual representation of this malicious behavior can be consulted in Figure 7.20.

Impersonator attack In this particular threat landscape, an AI agent becomes compromised due to the infiltration of malicious software, resulting in its display of aberrant behavior. Despite being incapacitated and rendered incapable of partaking in data collection or model training, this infected agent remains operational during the model dissemination phase, adhering to the standard sharing protocol. The fundamental distinction between a genuine agent and an impersonator lies in the deceptive capabilities of the latter. The impersonator possesses the artifice to masquerade the proficiency level

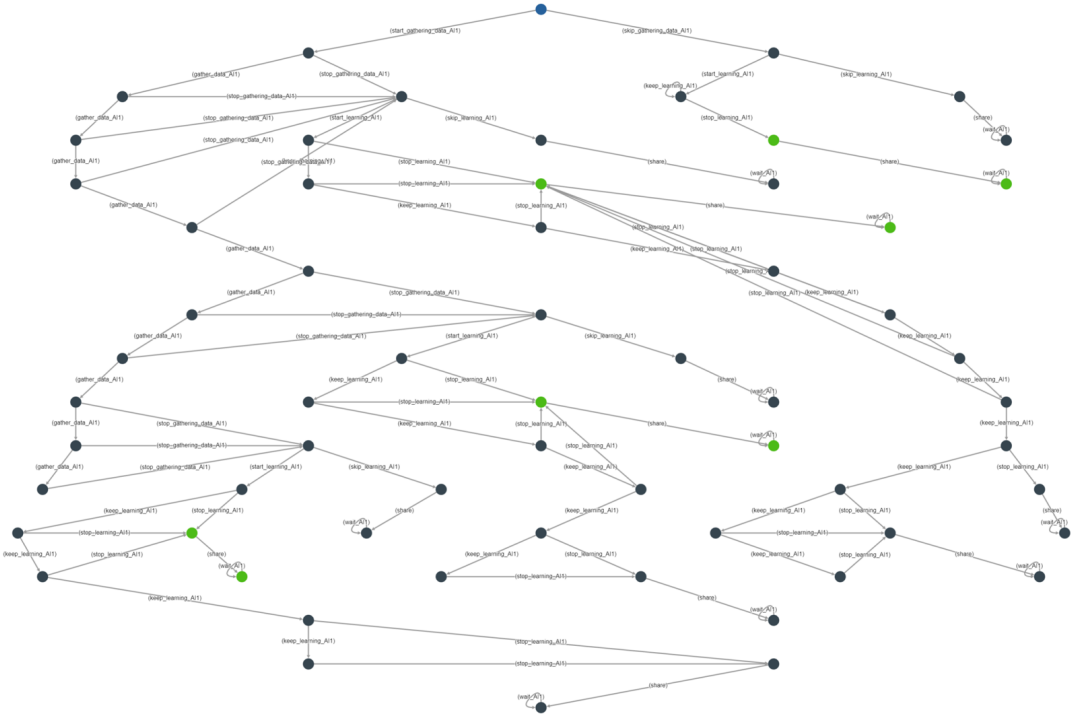

Figure 7.17: Model of SAI with one honest agent

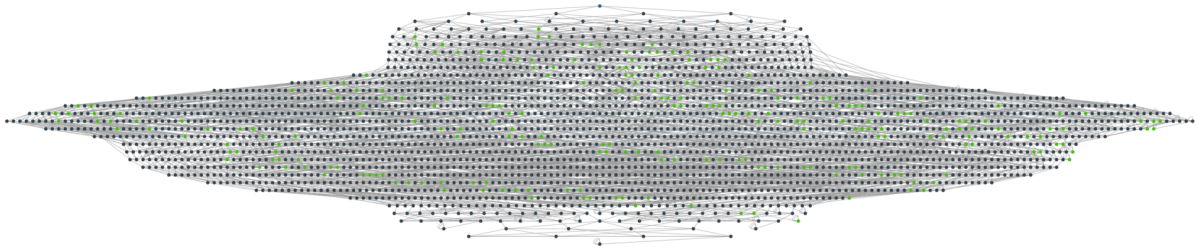

Figure 7.18: Model of SAI with two honest agents

of its AI model, thereby deceiving subsequent agents into mistakenly validating and assimilating it. Detailed insights into the modus operandi of an impersonator, as well as its graphical manifestation, can be gleaned from Figures 7.21 and 7.22, respectively.

7.5 Simple Models of Voting

Modelling voting protocols presents unique challenges due to the intricate interplay of strategic behavior, information asymmetry, and security requirements. In the present

```
Agent Mim:
init: start
shared share_1_with_mim: start -> start
    [Mim_model_quality=AI1_model_quality]

shared share_mim_with_1: start -> start

shared share_2_with_mim: start -> start
    [Mim_model_quality=AI2_model_quality]

shared share_mim_with_2: start -> start
```

Figure 7.19: Specification of the Man in the Middle agent

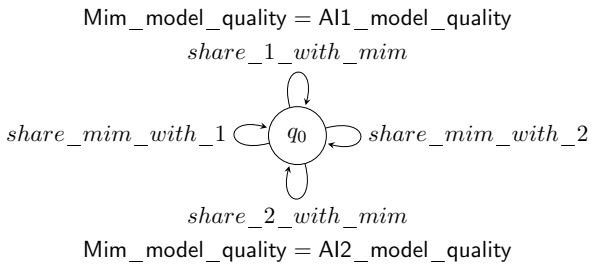

Figure 7.20: Graphical representation of the Man in the Middle agent

world of electronic and remote voting systems, accurately capturing these dynamics is essential for ensuring the integrity and trustworthiness of electoral processes. This section explores the formal modelling of simple voting scenarios, focusing on the representation of voter strategies, coercion resistance, and the epistemic states of various agents within the voting ecosystem.

7.5.1 Simple voting

We formalize a canonical voting-coercion interaction scenario involving two rational agents:

- Voter v with action space $\mathcal{A}_v = \{vote_1, vote_2\}$, representing selection between two candidates ($n = 2$)
- Coercer c with action space $\mathcal{A}_c = \{pun, np\}$, where pun signifies punitive action

```
Agent AI2:
init: start
set_quality_0: start -> set_quality
                [AI2_model_quality=0]

set_quality_1: start -> set_quality
                [AI2_model_quality=1]

set_quality_2: start -> set_quality
                [AI2_model_quality=2]

shared share_2_with_3: set_quality -> sharing
shared share_1_with_2: sharing -> start
```

Figure 7.21: Specification of the Impersonator agent

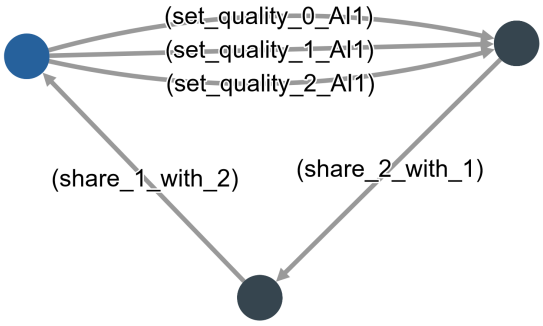

Figure 7.22: Graphical representation of Impersonator in STV

The game unfolds in three sequential phases:

1. Voting phase: Voter chooses candidate i through atomic action $vote_i$
2. Evidence phase: Voter selects between *give* (submitting verifiable voting proof) or *ng* (refusing proof transmission)
3. Enforcement phase: Coercer implements punitive strategy *pun* or maintains non-punishment *np*

The corresponding iCGS $\mathcal{M}_{\text{vote}}$ depicted in Figure 7.23 incorporates:

- Atomic propositions $\{\mathsf{vote}_1, \mathsf{vote}_2\}$ tracking voting outcomes
- Punishment indicator pun marking terminal states with coercive penalties

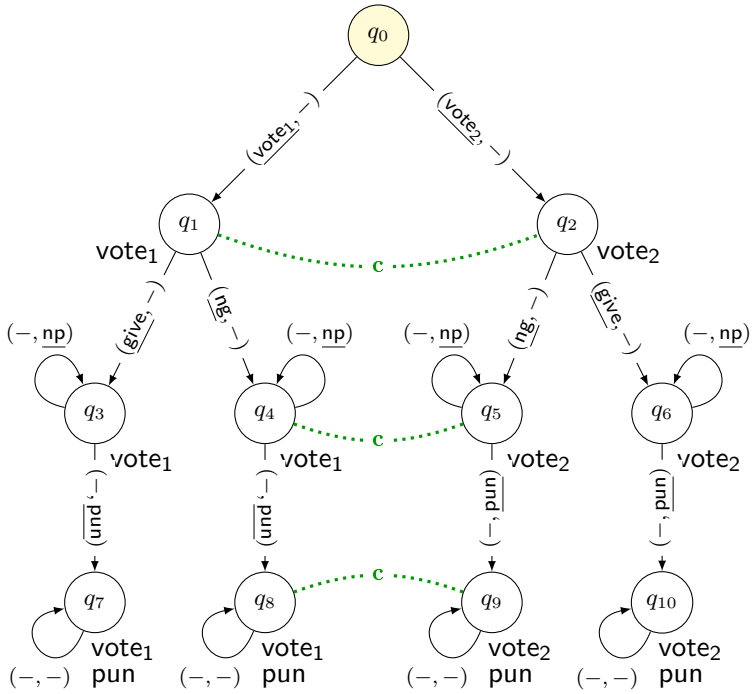

Figure 7.23: A simple model of voting and coercion

- Epistemic uncertainty represented by coercer's indistinguishability relation $\sim_c$ (dotted lines), capturing incomplete information about voter's actual choice

7.6 Selene

This section formalizes the modeling framework for the Selene e-voting protocol. We commence with a comprehensive protocol description, followed by a rigorous high-level abstraction of its theoretical underpinnings. The technical exposition subsequently transitions to a formalized model in Section 7.6.2, where implementation-specific details and security guarantees are systematically analyzed. The models are constructed under asynchronous semantics, reflecting the protocol's inherent operational dynamics. The global model is synthesized from local components, ensuring fidelity to the protocol's distributed architecture.

7.6.1 Outline of Selene

Selene Ryan et al. (2016) has been proposed as a recent electronic voting protocol specifically designed for low-coercion settings. Its cryptographic framework enables voters to demonstrably comply with coercer demands while preserving ballot secrecy. A key innovation lies in the protocol's ability to fully abstract the cryptographic operations from the voter's interaction, requiring no direct engagement with cryptographic primitives during voting.

The protocol operates through the following structured phases: First, the Election Authority initializes the system by generating election-specific cryptographic keys and constructing individualized vote trackers for each registered voter. These trackers undergo encryption and are subsequently anonymized via mixnet re-encryption techniques to eliminate voter-tracker associations. The anonymized tracker pool is then published on the publicly accessible Web Bulletin Board (WBB), while the authoritative assignment of trackers to voters remains confidential within the Election Authority's records. This separation ensures transparency of the cryptographic process without compromising voter anonymity.

During the voting phase, each voter completes, encrypts, and digitally signs her ballot, which is subsequently submitted to the election infrastructure. Following multiple cryptographic processing stages, the system publishes a tuple $(Vote_v, tr_v)$ for every $v \in Voters$ on the WBB, where $Vote_v$ and tr_v represent the decrypted ballot and corresponding tracker associated with voter v, respectively. Notably, voters remain unaware of their assigned tracker identifiers throughout this process. The publication of all cast votes in plaintext form on the WBBenables comprehensive public verification of the election process while maintaining the separation between voter identities and their cryptographic artifacts.

The post-election phase comprises the tracker number notification process. In the absence of coercion, voters submit requests for their α_v cryptographic proof to retrieve their authentic tracker tr_v. When coercive influence is exerted, voters instead communicate the coercer's mandated ballot specification to the election server. The system then generates a α'_v verifiable credential that cryptographically binds to the public commitment of record while mapping to a tracker associated with a vote complying with the coercer's demands. This mechanism ensures coercion resistance by guaranteeing that

the fake tracker corresponds to a valid vote already present in the anonymized tracker pool on the WBB, thereby preventing detection of the deception by the coercer.

Given our abstraction from cryptographic specifics in this analysis, we briefly note that Selene employs the ElGamal cryptosystem for asymmetric encryption. The protocol incorporates multiplicative homomorphic properties to enable vote aggregation without decryption, and utilizes computationally efficient Non-Interactive Zero-Knowledge (NIZK) proofs of knowledge to guarantee verifiable integrity of all cryptographic transformations executed on the Web Bulletin Board (WBB). These mechanisms ensure end-to-end verifiability while maintaining voter privacy.

An Abstract View of the Protocol

We present a methodological framework for characterizing the conceptual architecture of Selene at an abstract level. Social choice theory traditionally defines collective decision-making as a mathematical function mapping a set of voter preferences to a societal outcome. This function can be conceptually decomposed into two sequential mappings: (1) the establishment of individual voter choices and (2) the aggregation of these choices into a collective decision. End-to-end voter-verifiable protocols aim to ensure individual verifiability for the first mapping (allowing each voter to confirm their personal contribution) and universal verifiability for the second mapping (enabling public validation of the aggregation process). Conversely, coercion-resistant protocols prioritize ballot secrecy by maintaining cryptographic confidentiality of voter choices from all external entities while preserving voter control. Selene harmonizes these dual objectives through a novel architectural decomposition that separates voter identification from ballot content via cryptographic trackers, thereby creating distinct verifiability and privacy guarantees for each mapping component.

We formalize the foundational structure of Selene through set-theoretic constructs. Let $\mathcal{V}$ (voters), $\mathcal{T}$ (trackers), and $\mathcal{C}$ (candidates) denote finite sets with $|\mathcal{V}| = |\mathcal{T}|$. The protocol initiates with a cryptographically secure secret mapping $\sigma \in \Sigma$, where Σ represents the symmetric group of all bijections $\sigma : \mathcal{V} \to \mathcal{T}$. This σ establishes a one-to-one correspondence between voters and unique trackers. Subsequently, the voting process culminates in a public bulletin board function $\mathcal{B} : \mathcal{T} \to \mathcal{C}$, transparently recording each tracker's associated vote. The composition $\chi = \mathcal{B} \circ \sigma : \mathcal{V} \to \mathcal{C}$ defines the secret choice function linking voters to their selected candidates. Notably, the Election Authority maintains exclusive access to this end-to-end mapping χ, ensuring ballot secrecy

while preserving auditability through the public $\mathcal{B}$. This formalism provides a general framework applicable to any tracker-based voting system, separating identification from content verification through cryptographic permutation.

Combinatorial Aspects

Formally, define $\mathcal{B}$, χ, and σ as the public bulletin function, choice function, and cryptographic tracker bijection respectively. The coercer's epistemic uncertainty is quantified by analyzing the indistinguishability of alternative tracker assignments under observable vote outcomes. For $\tau, \tau' \in \mathcal{T}$, define the vote equivalence relation $\tau \sim_{\mathcal{B}} \tau'$ iff $\mathcal{B}(\tau) = \mathcal{B}(\tau')$, establishing tracker indistinguishability through identical vote records. Let $\mathcal{T}_{\mathcal{B}} = \mathcal{T}/\sim_{\mathcal{B}}$ denote the quotient set of tracker equivalence classes induced by $\mathcal{B}$. The coercer's uncertainty space corresponds to the automorphism group $\mathbf{Aut}(\mathcal{T}_{\mathcal{B}})$, whose cardinality reflects the number of plausible tracker-vote permutations consistent with the public record. This formal measure captures the protocol's coercion resistance as a function of the entropy inherent in the tracker assignment mechanism.

We formalize the coercer's uncertainty space through permutation group analysis. Let $\mathfrak{S}_{\mathcal{T}}$ denote the symmetric group on trackers $\mathcal{T}$, and define $\Gamma_{\mathcal{B}} = \{\gamma \in \mathfrak{S}_{\mathcal{T}} \mid \mathcal{B} \circ \gamma = \mathcal{B}\}$ as the stabilizer subgroup preserving vote equivalence under $\mathcal{B}$. For any $\gamma \in \Gamma_{\mathcal{B}}$, the invariance $\chi = \mathcal{B} \circ (\gamma \circ \sigma)$ ensures that observable vote distributions remain unchanged while concealing individual voter-trackers links. The cardinality $|\Gamma_{\mathcal{B}}|$ quantifies coercion resistance in single-election scenarios, representing the number of indistinguishable tracker permutations compatible with the public record. This measure directly relates to the entropy of the tracker assignment mechanism, where larger stabilizer subgroups imply stronger guarantees against coercion inference.

Definition 7.6.1 (Anti-coercion Space) *The* anti-coercion space $\mathcal{E}(\mathcal{B})$ *of an election with public bulletin function* $\mathcal{B}$ *is defined as the orbit* $\mathcal{E}(\mathcal{B}) = \{\mathcal{B} \circ \gamma \mid \gamma \in \mathfrak{S}_{\mathcal{T}}\}$ *under the action of the symmetric group* $\mathfrak{S}_{\mathcal{T}}$*. This space characterizes the set of all choice functions* χ' *that produce identical vote distributions to the true choice function* χ *when projected through* $\mathcal{B}$*.*

The cardinality of $\mathcal{E}(\mathcal{B})$ directly correlates with the computational indistinguishability guarantees against coercion attacks. Specifically, $|\mathcal{E}(\mathcal{B})| = |\Gamma_{\mathcal{B}}|$ grows combinatorially with the number of vote equivalence classes in $\mathcal{T}_{\mathcal{B}}$, becoming exponentially large except in degenerate cases (e.g., single-candidate dominance or minimal electorate sizes).

While this expansive space effectively obscures voter-trackers associations from coercers, it introduces significant computational verification challenges through state-space explosion. This inherent trade-off between coercion resistance entropy and verification scalability necessitates optimized group-theoretic verification algorithms for practical implementation.

7.6.2 Multi-Agent Model of SELENE

We present a formal multi-agent model of SELENE using *imperfect information concurrent game structures (iCGS)*, a well-established framework synthesizing concepts from temporal logic and game theory. These structures provide a unified foundation that generalizes both distributed system models (e.g., transition networks and finite-state machines) and game-theoretic constructs such as normal form, repeated, and extensive form games. The key advantage of iCGS lies in enabling rigorous definitions of *strategic play* and *strategic ability* within the system.

The system comprises a set *Voters* of voter agents, a single coercer *Coercer*, the Election Defense System *ElectionDS*, and an environmental agent *Environment*. We denote the complete agent set as *Agents*. Local states are determined by each agent's private variables, while global states represent full valuations of all agents' variables. Observational capabilities are defined such that each agent accesses its private variables and selected environmental parameters. This simplification excludes scenarios involving conflicting coercion demands from multiple coercers, a topic reserved for subsequent research.

The model is parameterized by the following natural numbers:

- n voters;
- k possible choices (i.e., the ways that a ballot can be filled);
- *maxCoerced* voters that can be influenced by the coercer;
- *votingWaitTime* and *helpRequestTime* that reflect the maximal number of steps the system waits for votes and notifications about being coerced, respectively.

We denote such model by $\mathcal{M}(n, k, maxCoerced, votingWaitTime, helpRequestTime)$.

In what follows, we omit auxiliary variables and actions that are not relevant to understanding the interplay between agents.

Variables

- *vote*: 0...k
- *demandedVote*: 0...k

Actions

- $Vote_i$, for $i \in \{1, \ldots, k\}$
- $INeedVote_i$, for $i \in \{1, \ldots, k\}$
- *FetchGoodTracker*
- *CopyRealTracker*
- *Wait*
- *Finish*

Can Observe

- WBB: *public election function*
- *elections' stage (init/voting/defense)*
- *his real tracker (when permitted)*
- *his exposed tracker*

Can Set

- *his exposed tracker (via CopyRealTracker)*

Figure 7.24: A Voter agent

Agent *Environment*

The purpose of the *Environment* agent is twofold. Firstly, it serves as a container for variables shared by selected agents. The agents can have read-only or write-only access to the variables (denoted by *Can observe* and *Can set*, respectively, in agent interfaces in Figures 7.24, 7.25, and 7.26). Secondly, it traces the passage of time and changes the stage of elections. Namely, the elections start in the **initial stage**, when the secret bijection is non-deterministically prepared. Then, the **voting phase** is open and the clock is started. This phase ends when either all the voters send their choices or time exceeds *votingWaitTime*. Then, the system enters the **defense stage** and the clock restarts. The defense stage ends either when the clock exceeds *helpRequestTime* or all the voters execute the *Finish* action. Note that every agent can observe WBB, i.e., public election function, the stage, and the clock value. The clock limits are also public knowledge.

Voter agents

Each *Voter* shares the same structure, presented in Figure 7.24. It is able to record via the *vote* variable the vote cast for choice $i \in \{1, \ldots, k\}$ by executing the action $Vote_i$.

Variables

- $falseTrackerSentToVoter_i$: Boolean, for $i \in \{1, \ldots, n\}$

Actions

- $SetFalseTrackerOfVoter_i To_j$, for $1 \leq i \leq n$, $1 \leq j \leq k$
- *Wait*

Can Observe

- WBB: *public election function*
- *the secret bijection*
- *elections' stage (init/voting/defense)*

Can Set

- *the exposed tracker of every Voter*

Figure 7.25: The *ElectionDS* agent

This action can be used only once, in the voting phase. It also records the coercer's request to vote for *demandedVote*. In both the cases 0 denotes that the variable is not set, i.e. the agent did not vote yet and has not been contacted by the coercer, respectively. In addition to the public variables of *Environment*, each *Voter* can observe his real tracker, obtained in the defense phase by executing action *FetchGoodTracker*. The agent can also observe his exposed tracker, i.e., the number assigned by *ElectionDS*, as presented to the *Coercer* agent. This becomes possible after requesting in the defense phase a tracker that points to a specific choice $i \in \{1, \ldots, k\}$, by firing action $INeedVote_i$. After obtaining his real tracker a *Voter* can decide to make it visible to the coercer by executing action *CopyRealTracker*. Finally, the agent can always *Wait*, unless the clock reaches the limit set for a phase. In the latter case, if it is the voting phase, then the *Voter* needs to decide on the vote immediately, and if it is the defense phase, then it automatically ends his participation by firing action *Finish*. It should be noted that these actions are autonomous, e.g., a *Voter* can signal *ElectionDS* that he is coerced to vote in a selected way, even if coercion does not take place.

Variables

- $voteDemandedFromVoter_i$: $0, \ldots, k$, for $i \in \{1, \ldots, n\}$

Actions

- $ReqVote_iFromVoter_j$, for $1 \leq i \leq k$, $1 \leq j \leq n$
- *Wait*

Can Observe

- WBB: *public election function*
- *elections' stage (init/voting/defense)*
- *the exposed tracker of every Voter*

Figure 7.26: The *Coercer* agent

Agent *ElectionDS*

The structure of *ElectionDS* agent is presented in Figure 7.25. The agent can, in addition to the public variables of *Environment*, observe the secret bijection function. This gives *ElectionDS* the full knowledge of the secret election function. The boolean variables $falseTrackerSentToVoter_i$ record that a voter $i \in \{1, \ldots, n\}$ requested and has been provided with a false tracker. This request is fulfilled by executing an action $SetFalseTrackerOfVoter_iTo_j$ that sets the exposed tracker of voter $1 \leq i \leq n$ to choice $1 \leq j \leq k$. Note that while *ElectionDS* can set the value of the exposed tracker of any *Voter*, it cannot read the current value of the variable. Therefore, each *Voter* can first request a false tracker pointing to any choice and expose his real tracker afterwards, unknowingly to *ElectionDS*. Finally, *ElectionDS* can always *Wait*.

Agent *Coercer*

The structure of *Coercer* is presented in Figure 7.26. Starting from the initial phase until the votes are published, the agent can demand from any voter $1 \leq j \leq n$ to vote for $1 \leq i \leq k$, by executing $ReqVote_iFromVoter_j$ action. Such a request can be made at most once per voter and the total number of requests cannot exceed *maxCoerced*. These choices are recorded using variables $voteDemandedFromVoter_i$, where $1 \leq i \leq n$. As previously, the value of 0 signifies that no request has been made. The agent can

```
Agent Coercer

Lobsvars = {exposedTrackerOfVoter1, exposedTrackerOfVoter2};

Vars:
    coercedVoters: 0..2;
    voteDemandedFromVoter1: 0..2;
    voteDemandedFromVoter2: 0..2;
end Vars

Actions = {ReqVote1FromVoter1, ReqVote2FromVoter1,
           ReqVote1FromVoter2, ReqVote2FromVoter2, Wait};

Protocol:
    coercedVoters < maxCoerced and voteDemandedFromVoter1 = 0
    and Environment.votesPublished = false:
    {ReqVote1FromVoter1, ReqVote2FromVoter1, Wait};

    coercedVoters < maxCoerced and voteDemandedFromVoter2 = 0
    and Environment.votesPublished = false:
    {ReqVote1FromVoter2, ReqVote2FromVoter2, Wait};

    Other: {Wait};
end Protocol

Evolution:
    coercedVoters = coercedVoters + 1 if
    (Action = ReqVote1FromVoter1
    or Action = ReqVote2FromVoter1
    or Action = ReqVote1FromVoter2
    or Action = ReqVote2FromVoter2);

    voteDemandedFromVoter1 = 1 if Action = ReqVote1FromVoter1;
    voteDemandedFromVoter1 = 2 if Action = ReqVote2FromVoter1;
    voteDemandedFromVoter2 = 1 if Action = ReqVote1FromVoter2;
    voteDemandedFromVoter2 = 2 if Action = ReqVote2FromVoter2;
end Evolution

end Agent
```

Figure 7.27: ISPL code of the *Coercer* agent

observe all public variables of *Environment* and all the exposed trackers of all voters. At any step, the *Coercer* agent can *Wait*.

Atomic propositions

In order to construct formulae that can be interpreted in the model, we need some atomic propositions. We set $Props = \{\mathsf{finished}\} \cup \{\mathsf{vote_{v,i}} \mid 1 \leq v \leq n, 1 \leq i \leq k\}$. Proposition $\mathsf{finished}$ denotes that the execution of the protocol has come to an end, and it holds iff all the voters have executed *Finish* or the clock has exceeded *helpRequestTime*. Formula $\mathsf{vote_{v,i}}$ says that voter v has voted for candidate i; it holds iff v's variable *vote* contains i.

7.6.3 Implementation of the Model

In Figure 7.27, we present the ISPL code implementing the *Coercer* agent. The local variables of the agent are denoted by *Vars*, *Lobsvars* denotes the set of the environment

variables that the agent can observe, and *Actions* are action labels. The *protocol* section specifies which actions are available at what states; the *evolution* section defines the consequences of their execution.

7.7 Specification of Voting Properties

Formalizing Voter Verifiability in ATL

The Alternating-time Temporal Logic (**ATL**) framework enables precise specification of voting system properties. A key requirement is *voter-verifiability*, which ensures voters can confirm their votes' correct handling through the Web Bulletin Board (WBB). This verification process, denoted $checkWBB$, generates two possible outcomes:

$$\text{Initial specification:} \quad \langle\langle voter \rangle\rangle \mathbf{F}(\mathsf{checkWBB\_ok} \vee \mathsf{error})$$

$$\text{Improved specification:} \quad \langle\langle voter \rangle\rangle \mathbf{F}(\mathsf{checkWBB\_ok} \vee \mathsf{checkWBB\_fail})$$

The initial specification contains a critical flaw: it permits trivial satisfaction when the voter unconditionally signals error without performing $checkWBB$. The refined formulation enforces:

- Mandatory execution of $checkWBB$ verification protocol
- Guaranteed termination in either checkWBB_ok (vote correctly recorded) or checkWBB_fail (discrepancy detected)
- Strategic capability of the voter to achieve conclusive verification outcomes

This distinction ensures proper modeling of verifiability as a two-phase process: first executing the verification protocol, then transitioning to a definitive epistemic state about the vote's status on the WBB.

Formalizing Dispute Resolution Requirements

Building upon verification capabilities, we extend the logical framework to address dispute resolution through the ATL formula:

$$\mathsf{AG}\,(\mathsf{checkWBB\_fail} \rightarrow \langle\langle v \rangle\rangle \mathbf{F}\mathsf{error})$$

This expression formalizes the *error-signaling* property: whenever a verification failure occurs (checkWBB_fail), the voter maintains a strategic ability to initiate dispute protocols (error) along all possible execution paths (**AG**).

A comprehensive characterization of dispute resolution necessitates significantly enhanced modeling granularity:

- Evidence submission mechanisms to institutional authorities (electoral commissions, judicial bodies)
- Formal deliberation processes including evidence evaluation and decision derivation
- Enforcement protocols for corrective actions (election annulment, revotes, legal penalties)

Conjecture 7.7.1 *Dispute resolution requirements in coercive voting environments demand:*

1. *Structurally richer ICGS models incorporating institutional actors and legal processes*
2. *Temporal-epistemic specifications ensuring irrevocable evidence binding*
3. *Strategic capabilities for voters to navigate multi-stage dispute workflows*

This represents a strictly greater cognitive and strategic burden compared to basic verifiability, requiring voters to execute purposeful, potentially non-myopic action sequences under adversarial conditions.

Strategic-Epistemic Verification Framework

The propositional-level specifications discussed previously depend on explicit state labeling with verification outcomes ($\mathsf{checkWBB\_ok}, \mathsf{checkWBB\_fail}$). To capture higher-level epistemic requirements, we extend **ATL** with knowledge operators K_a following standard epistemic logic conventions. For agent a, formula $K_a\varphi$ asserts that a knows φ holds in the current state.

Definition 7.7.2 (Epistemic Verification Property) *Voter-verifiability as a strategic-epistemic property requires the voter v to possess a strategy ensuring complete epistemic closure about vote registration. Formally:*

$$\langle\langle v\rangle\rangle \mathbf{F} \bigwedge_{i \in Cand} (K_v \mathsf{voted_i} \vee K_v \neg \mathsf{voted_i})$$

This formula specifies that voter v can execute a strategy guaranteeing eventual knowledge of either $\mathsf{voted_i}$ *(vote correctly recorded for candidate i) or its negation for all candidates in Cand.*

Remark 7.7.3 *The epistemic formulation avoids the verification outcome labeling problem in propositional approaches. By directly quantifying over the voter's knowledge states, it ensures non-trivial verification through proper execution of checkWBB rather than passive proposition signaling.*

Formalizing Receipt-Freeness via Strategic-Epistemic Logic

Receipt-freeness represents a critical security requirement in coercion-resistant voting systems, formalized through strategic-epistemic properties in extended ATL frameworks. The foundational definition asserts:

Definition 7.7.4 (Strong Receipt-Freeness) *The system must prevent voters from producing credible evidence of their vote choice to coercers. This is expressed as:*

$$\bigwedge_{i \in Cand} \neg\langle\langle c, v\rangle\rangle \mathbf{G}\,(\mathsf{end} \rightarrow (K_c \mathsf{vote_i} \vee K_c \neg \mathsf{vote_i}))$$

where c denotes the coercer, v the voter, and Cand the candidate set.

Remark 7.7.5 (Alternative Characterizations) *A weaker yet practically significant formulation focuses on coercer's inability to detect voter non-compliance:*

$$\bigwedge_{i \in Cand} \neg\langle\langle c, v\rangle\rangle \mathbf{G}\,(\mathsf{end} \wedge \neg \mathsf{vote_i} \rightarrow K_c \neg \mathsf{vote_i})$$

This captures scenarios where the coercer only needs to verify whether the voter deviated from a prescribed choice.

Proposition 7.7.6 (Vote Anonymity Specification) *The system guarantees vote anonymity if for all non-voter agents a and candidates i:*

$$\mathbf{AG}\,(\neg K_a \mathsf{vote_i} \wedge \neg K_a \neg \mathsf{vote_i})$$

This ensures no external agent can infer individual vote content at any execution path.

While strategic-epistemic specifications provide expressive power, they introduce semantic challenges in ATL extensions Ågotnes et al. (2015); Jamroga and van der Hoek (2004). Following our methodological approach, we prioritize strategic-operator formulations that avoid epistemic quantification complexities while maintaining behavioral fidelity.

7.8 Random Models

In multi-agent system development, researchers often focus on modeling real-world processes, games, or narratively coherent scenarios. However, in specific contexts, the structural properties of the model itself take precedence over application-specific relevance. This includes cases where structural constraints are required for algorithmic optimization - such as model checking techniques that are optimized for specific topological features - or when requiring a scalable, customizable benchmark framework for algorithm validation. The following section introduces a synthetic, structurally arbitrary model devoid of contextual narrative, designed specifically to enable systematic evaluation of computational methods through precise control over topological parameters.

The synthetic environment represents a single rational agent interacting with a non-deterministic environment modeled as stochastic transitions. Model generation follows a structured procedure:

1. *Graph Construction*: We initialize a directed acyclic graph (DAG) through path-based node allocation. The state space cardinality is determined by the parameter $N \in \mathbb{N}$, where N corresponds to the total number of states. Initial path structures are established via random edge assignment between sequential node clusters, followed by augmentation with cross-path edges sampled from uniform distribution across non-adjacent node pairs.

2. *Transition Formalization*: Each node $s_i \in S$ generates a transition relation $T(s_i)$ through combinatorial selection: For all outgoing edges $E_{out}(s_i)$, we randomly sample k-element subsets (where $k \leq |E_{out}(s_i)|$) to construct partial transition functions. Action labels $\alpha \in \mathcal{A}$ are assigned through bijection with sampled edge subsets, ensuring distinct action definitions per state.

3. *Epistemic Structuring*: The state space S undergoes partitioning into epistemic equivalence classes $\mathcal{K} = \{K_1, K_2, ..., K_m\}$ via random equivalence relations satisfying:

$$\forall K_i, K_j \in \mathcal{K},\ (K_i \cap K_j \neq \emptyset) \Rightarrow (K_i = K_j) \quad \text{and} \quad \bigcup_{K \in \mathcal{K}} K = S$$

This partitioning maintains class cardinality constraints $|\mathcal{K}| = f_{\text{epist}}(N)$, where f_{epist} is a sublinear function of state count.

4. *Objective Specification*: Winning states $W \subset S$ are designated as terminal states in each path, selected through uniform random sampling from path-specific terminal nodes. The reward structure satisfies $|W| = f_{\text{win}}(N)$ with f_{win} representing configurable density parameters.

All complexity metrics (connection density, action space dimensionality, epistemic class count) are governed by parametric functions of N, enabling systematic variation of computational requirements while maintaining formal verifiability through controlled topological properties.

7.9 Other Models

In this section we briefly introduce interesting models that appear in various publications. We will refer to them in further sections.

7.9.1 Castles Benchmark

The *Castles* model have been proposed in Pilecki et al. (2014). The model consists of one agent called *Environment* that keeps track of the health points of three castles, plus a number of agents called *Workers* each of whom works for the benefit of a castle. Each castle is assigned a certain number of health points (HP, ranging from 0 to 3), that represent the current condition of the castle. When castle's HP drops to 0 it means that the castle is *defeated*.

Workers can execute the following actions:

- $attack_c$: attack a castle c, which is one of the castles they do not work for,
- $defend$: defend the castle they do work for,
- $idle$: do nothing.

Doing nothing is the only available action to a Worker of a defeated castle. No agent can defend its castle twice in a row, it must wait one step before being able to defend again. A castle gets damaged if the number of attackers is greater than the number of defenders, and the damage is equal to the difference. In the initial state, all castles have 3 HP and every Worker can engage in defending its castle. Every Worker knows if it can currently engage in defending its castle, and can observe for each castle if it

is defeated or not, which defines the indistinguishability relation for the agents. The model is parameterized by the number of agents and the allocation of Workers. For example, an instance with 1 worker assigned to the first castle, 3 workers assigned to the second and 4 to the third castle will be denoted by *9 (1,3,4)*.

7.9.2 TianJi Model

The TianJi model has been proposed in Busard et al. (2014). The model consists of two agents: *Tian Ji* and the *King*. Each agent has n horses numbered $1, \ldots, n$. In the game, Tian Ji and the King send their horses one by one against each other. Horse i of Tian Ji wins the race with King's horse j iff $i > j$. At each stage, the agents know the current score and their own remaining horses, but not those of the opponent. Moreover, the decisions at each round are made simultaneously, so one does not know which horse is currently sent by the other player. The player whose horses won most races wins the game.

7.10 Challenges and Lessons Learnt

Modeling Complex Systems The formal verification of real-world systems revealed three fundamental challenges:

1. **Abstraction-Completeness Trade-off**: Effective modeling requires balancing abstraction with semantic fidelity. The bridge card game analysis demonstrated how over-abstraction (e.g., omitting trick resolution dynamics) could compromise verification of strategic reasoning under partial observability, while excessive detail in drone coordination models led to state-space explosion in systems with $N > 10$ agents.

2. **Temporal-Epistemic Complexity**: The SAI framework highlighted the exponential growth of epistemic equivalence classes. This necessitated optimized group-theoretic verification algorithms to maintain computational tractability.

3. **Security-Centric Modeling**: Coercion-resistant voting protocols like Selene required novel formalizations of strategic-epistemic properties. The implementation of $K_v(\text{voted}_i \vee \neg\text{voted}_i)$ specifications demanded sophisticated state-space partitioning to prevent trivial satisfaction through passive proposition signaling.

Verification Methodology Insights Our multi-agent system analyses yielded key methodological advancements:

1. **Structured Abstraction Frameworks**: The bridge endplay model established best practices for state-space reduction through:
   - Hierarchical state representation with $(\text{hands}, \text{tricks}, \text{board}, \text{clock})$ tuples
   - Contextual goal structures separating verification objectives from environmental dynamics
2. **Parametric Scalability Mechanisms**: Drone coordination models demonstrated the effectiveness of sublinear complexity functions $f_{\text{epist}}(N)$ and $f_{\text{win}}(N)$ in managing state-space explosion. This enabled verification of systems through controlled topological parameterization.
3. **Security-by-Design Principles**: The Selene analysis emphasized the importance of integrating security properties at the modeling stage. Key innovations included:
   - Cryptographic commitment tracking through SN $\rightarrow$ WBB mappings
   - Robust anti-coercion spaces $E(B)$ with combinatorial indistinguishability guarantees

8

STV: StraTegic Verifier

STV is a formal verification framework designed to rigorously assess agents' strategic capabilities in multi-agent systems, with a central objective of synthesizing memoryless strategies under imperfect information conditions that provably satisfy specified temporal goals. Its analytical scope comprises two orthogonal dimensions: (1) model checking of *functionality requirements*, formalized as the capacity of legitimate users to realize their intended objectives despite environmental uncertainties, and (2) verification of *security properties*, defined through the formal infeasibility of adversarial entities compromising system integrity. The framework explicitly addresses both cooperative and adversarial agent interactions through a unified specification methodology.

The content of this chapter is based on the following papers: Kurpiewski et al. (2019a, 2021).

8.1 Functionalities

The STV model checker performs explicit-state model checking, representing global states and transitions within the model explicitly in the verification process's memory. This representation is akin to a linked-list graph structure, where each state has an associated list of outgoing transitions. To enhance verification efficiency at the expense of increased memory usage, STV sorts not only forward transitions but also their reverse counterparts. This approach allows for quick computation of a state's pre-states by iterating over its transition list.

We now provide a detailed overview of the functionalities implemented in STV.

8.1.1 Implemented Algorithms

Approximate fixpoint verification. STV supports standard fixpoint verification for $\mathbf{ATL}_{\mathrm{Ir}}$ and approximate fixpoint verification for $\mathbf{ATL}_{\mathrm{ir}}$, employing an algorithm detailed in Section 3.1. An optimized version of this algorithm, outlined in Section 3.2, utilizes a Disjoint-Union data structure to represent epistemic class sets.

Depth-first strategy synthesis with removal of dominated strategies. The tool enables depth-first strategy synthesis for both $\mathbf{ATL}_{\mathrm{Ir}}$ and $\mathbf{ATL}_{\mathrm{ir}}$. Optimization occurs through the removal of dominated strategies, as described in Section 4.2.4. Additional heuristics have been implemented to further enhance the tool's verification capabilities.

Depth-bounded DFS strategy synthesis. STV supports strategy synthesis for perfect recall semantic versions of ATL, namely $\mathbf{ATL}_{\mathrm{IR}}$ and $\mathbf{ATL}_{\mathrm{iR}}$. Users can initiate this process by specifying the desired depth of the search tree, correlating to the length of the agent's action history under examination.

Partial-order reduction. The tool implements a fully automated reduction applicable only with asynchronous semantics, as described in Section 6.2. It requires a predefined set of propositions for the reduction process and generates a reduced model in place of the full model.

Bisimulation checking. The tool facilitates A-bisimilar checking between two models for a specified coalition A. In addition to defining both models, users must provide the bisimulation relation between corresponding states and the chosen coalition.

8.1.2 Model Specification Language

The STV model-checker enables model specification through a user-friendly text-based input method. Models are defined as Asynchronous Multi-Agent Systems (S), where agents are instantiated from a template. The specification for each agent encompasses:

1. **Local Variables and Initial Values**: Outlines the agent-specific variables and their initial states.
2. **Initial State**: Specifies the starting state of the agent.
3. **Transition System**: Detailed as a list, it contains the transitions that an agent can perform, classified as 'private' or 'shared':

- *Private Transitions*: Unique to the agent, not involving other agents.
- *Shared Transitions*: Involve interactions with other agents.

4. **Transition Specification**: Describes each transition in terms of:
   - *Source State*: The starting state of the transition.
   - *Target State*: The state achieved after the transition.
   - *Action*: The action that triggers the transition.
   - *Guard Condition*: A condition that must be met for the transition to occur.
   - *Post Condition*: The expected variable state changes post-transition.

The number following an agent's name in brackets indicates the number of instances to be created from the template. During file parsing, each 'aID' in the agent specification is replaced with a unique identifier, like 'Train1' or 'Train2'.

Model Specification Example: The example model includes two agent types: Train and Controller, instantiated as Train[2] and Controller[1]. Each agent type has its own transitions.

- **Agent Train[2]**:
  - *Initial State*: `wait`
  - *Shared Transitions*:
    - `a1_aID`: Transition from `wait` to `tunnel`, activated with `aID_in=true`.
    - `a2_aID`: Transition from `tunnel` to `away`, activated with `aID_in=false`.
    - `a3`: Private transition from `away` to `wait`.
- **Agent Controller[1]**:
  - *Initial State*: `green`
  - *Shared Transitions*:
    - `a1_Train1` and `a1_Train2`: Transition from `green` to `red`.
    - `a2_Train1` and `a2_Train2`: Transition from `red` to `green`.

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Figure 8.1: Train-Controller model specification in STV

Additional elements like `REDUCTION` and `COALITION` define inter-agent relations. The `FORMULA` states a logic formula for validation, focusing on the state transitions of trains managed by the controller.

The specification of the Train-Controller model in STV is illustrated in Figure 8.1.

Other specifications has been shown throughout the thesis, e.g., in Figures 7.21, 7.19, and 7.13.

8.1.3 Graphical Interface

The graphical interface of the STV tool is developed using the Electron framework, resulting in a user-friendly desktop application. For optimal functionality, the user is required to have Python installed on their computer, with version 3.7 or higher. The interface is designed to facilitate various tasks related to model generation, visualization, and verification. Examples of the interface in use are presented in the figures **??**, **??**, **??**, and **??**.

Model Loading Options. The tool provides users with the flexibility to either import a model description from a file or select from an array of predefined models. This feature enhances the tool's accessibility and ease of use.

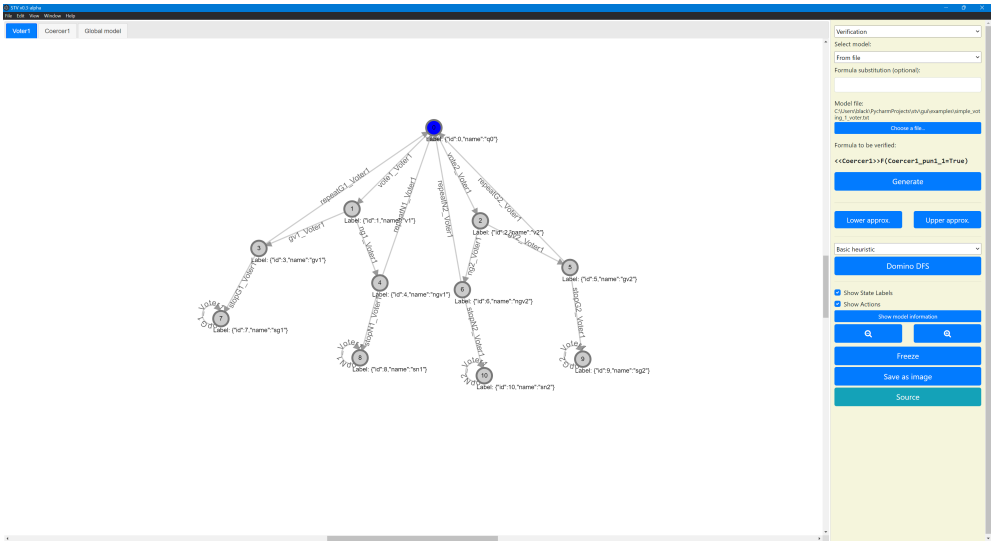

Figure 8.2: Graphical User Interface of the STV tool, local model view

Transition Graph Display. Upon selecting a model, it can be generated and visualized within the interface as a transition graph. This graphical representation is key to understanding the model's structure and dynamics.

Graph Customization and Interaction. The transition graph is automatically aligned for optimal viewing. However, users have the option to manually adjust the placement of nodes on the screen to suit their viewing preferences. Additionally, the interface includes zoom functionality to allow for both detailed examination and broader overview of the model.

Display Options. The interface offers options to show or hide the descriptions of transitions, which include user actions, and the descriptions of each state, encompassing the state name and values of internal variables.

Asynchronous Model Features. In cases where the user generates an asynchronous model, the tool displays not only the global model but also the local model templates. This dual-view feature provides a comprehensive understanding of the system's operations.

Model Reduction Visualization. For asynchronous models, the tool allows the generation of a reduced model. To illustrate the relationship between the global and reduced models, corresponding states are marked with the same color, enabling easy comparison and analysis.

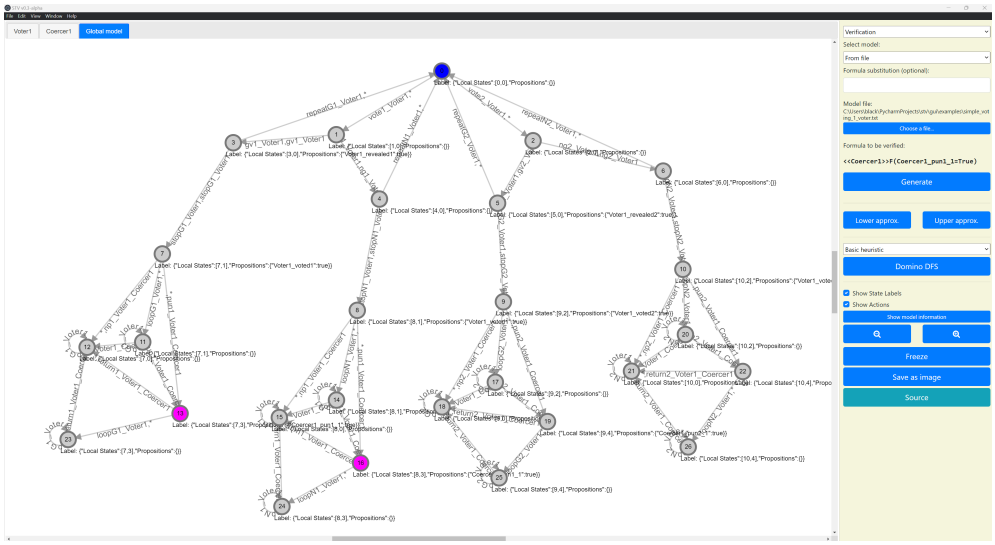

Figure 8.3: Graphical User Interface of the STV tool, global model view

Verification Algorithms Utilization. The tool supports the application of various verification algorithms. During the verification process, initial states and winning states are distinctly colored to enhance clarity. Upon completion, the verification results are displayed, with states reachable by the coalition's winning strategy highlighted in an additional color.

Bisimulation Checking Support. The tool's interface adeptly supports bisimulation checking by displaying two user-provided models side-by-side. Corresponding states in these models are marked with the same color to facilitate analysis of the bisimulation relation.

8.2 Implementation Details

The source code of STV has been structured in separate Python modules, which we describe below.

8.2.1 Logics

The logics module contains implementation of classes for representation of ATL models. The main class is ATLIrModel that represents the model with perfect information in form of a transition graph. Transitions are represented using the Transition class.

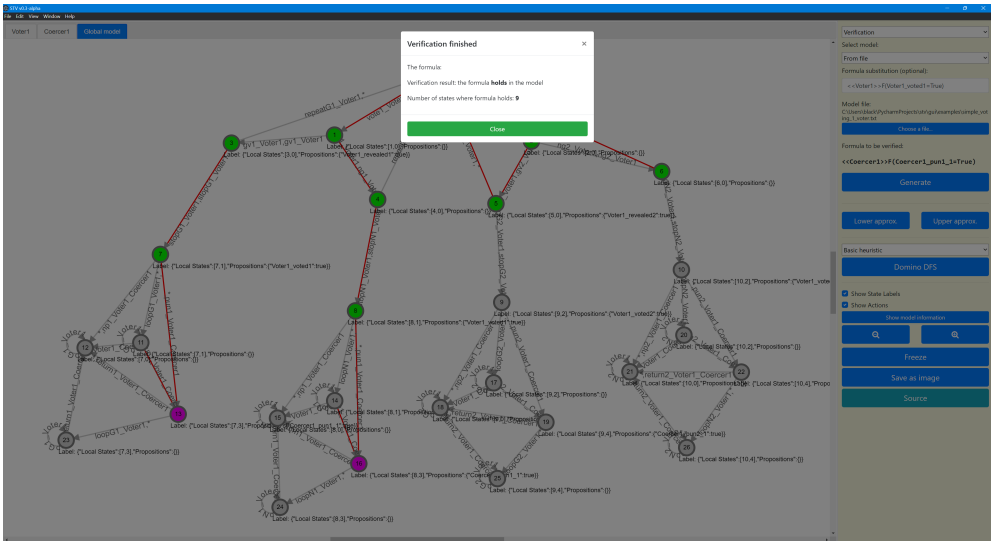

Figure 8.4: Graphical User Interface of the STV tool, verification result, strategy found

8.2.1.1 Transition

The class named `Transition` is designed to represent transitions in a model. This class encapsulates key properties of a transition, including the next state, associated actions, and time.

Properties and Validations. The `Transition` class comprises three main properties:

- `next_state`: An integer representing the identifier of the next state in the transition. It includes a validation check to ensure the value is non-negative.
- `actions`: A list of strings, each representing an action associated with the transition. This property is safeguarded by a validation that prohibits an empty list of actions.
- `time`: An integer indicating the time associated with the transition, with a validation to ensure it is not negative.

Initialization and String Representation. The class constructor (`__init__`) initializes a `Transition` instance with the next state identifier, a list of actions, and an optional time parameter (defaulting to 1). The class also includes two methods, `to_str` and `__str__`, for generating a string representation of the transition. These methods

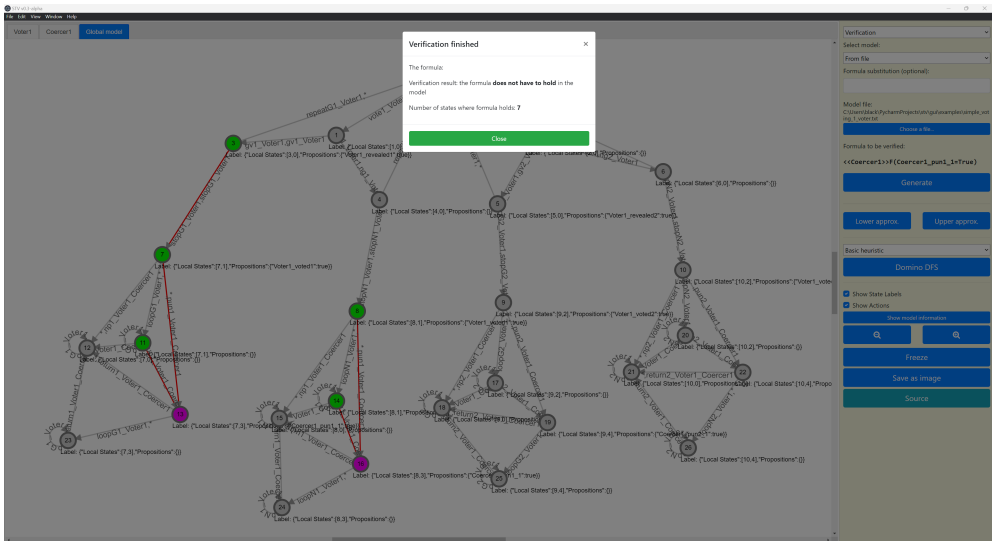

Figure 8.5: Graphical User Interface of the STV tool, verification result, strategy not found

provide a formatted output displaying the next state, actions, and time associated with a transition.

8.2.1.2 ATLIrModel

The ‘ATLIrModel‘ class is implemented to create ATL models with perfect information and imperfect recall. It encompasses several properties and methods essential for modeling and verifying strategies in multi-agent systems.

Class Properties: The class defines several properties:

- `number_of_agents`: Indicates the total number of agents in the model, with a validation to ensure it’s non-negative.
- `transitions` and `reverse_transitions`: Stores the forward and reverse transitions of the model.
- `agents_actions` and `pre_states`: Holds actions for each agent and pre-states for each state in the model.
- `number_of_states` and `states`: Represents the total number of states and the list of states in the model.

- `strategy`: Manages the strategic choices for each state.

Constructor and Initialization: The constructor accepts the number of agents and an optional parameter for opponent reactiveness. It initializes the model's states, transitions, and agent actions.

Methods for Strategy Synthesis and Verification: The class includes various methods for synthesizing and verifying strategies. These include methods for adding actions and transitions, DFS-based synthesis algorithms, and methods for computing the minimum and maximum fixpoint for one agent, many agents, or no agents.

Utility Methods: Additional methods like `prepare_result_states`, `marked_winning_states`, and `print_model` provide utilities for manipulating and displaying model states.

In summary, the 'ATLIrModel' class serves as a comprehensive framework for defining and analyzing ATL models with perfect information and imperfect recall, equipped with functionalities for detailed strategy synthesis and verification.

8.2.1.3 ATLirModel

The 'ATLirModel' class, inheriting from 'ATLIrModel', is designed for creating ATL models with imperfect information and imperfect recall. It includes additional properties and methods specific to handling epistemic relations and imperfect information scenarios.

Class Properties:

- `epistemic_class_membership`: Tracks the epistemic class membership for each agent in every state.
- `imperfect_information`: Represents the imperfect information available to each agent.
- `finish_model_called`: A flag to indicate whether the model construction process is complete.

Initialization: The constructor initializes the epistemic relations and sets up the model. It includes methods like `init_epistemic_relation` and `finish_model` to manage epistemic classes and finalize the model.

Epistemic Class Management:

- `add_epistemic_class`: Adds an epistemic class for a specific agent and updates the epistemic class membership.
- `epistemic_class_for_state_one_agent` and `epistemic_class_for_state_multiple_agents`: These methods compute the epistemic class for a given state and agent(s).

Strategy Synthesis and Verification: The class extends various methods for computing winning strategies in scenarios with imperfect information. This includes different versions of depth-first search (DFS) algorithms and basic formula computation for one or multiple agents.

DFS Algorithms:

- `dfs_alg_one_agent`, `dfs_counting_alg_one_agent`, `dfs_counting_bounded_alg_one_agent`, and others: These methods implement DFS-based strategy synthesis with different constraints and optimizations for handling imperfect information.

In summary, the 'ATLirModel' class provides a robust framework for modeling and analyzing ATL scenarios with imperfect information and recall. It extends the functionalities of the 'ATLIrModel' with additional considerations for epistemic classes and imperfect information, offering a wide range of methods for strategy synthesis and verification.

8.2.1.4 ATLirModelDisjoint

The 'ATLirModelDisjoint' class extends the 'ATLIrModel' to create ATL models with imperfect information and imperfect recall, utilizing a disjoint-union structure. This class enhances the handling of epistemic relations and state transitions within ATL models.

Key Properties:

- `epistemic_class_membership`: Manages the membership of states in epistemic classes for each agent.
- `epistemic_class_disjoint`: A list of disjoint-set data structures for managing epistemic classes.
- `imperfect_information`: Stores imperfect information available to each agent.

- `can_go_there`: A structure to determine the possible transitions for each state and action.
- `finish_model_called`: Indicates if the model construction is completed.

Constructor and Initialization: The constructor initializes the epistemic relations using disjoint sets. It sets up the structures for managing imperfect information and possible state transitions.

Epistemic Class Management:

- `add_epistemic_class`: Adds an epistemic class for an agent and updates the disjoint-set structure.
- `find_where_can_go`: Determines the possible states that can be reached for each action in an epistemic class.

Formula Computation: The class includes methods like `minimum_formula_one_agent` and `basic_formula_one_agent` to compute winning strategies, utilizing the disjoint-set structure for efficient management of state relations.

Utility Methods: Additional methods like `is_reachable_by_agent`, `epistemic_class_for_stat` and `epistemic_class_for_state_multiple_agents` provide essential functionalities for strategy computation and epistemic relation management.

In essence, the 'ATLirModelDisjoint' class provides advanced functionalities for ATL models with imperfect information and recall, leveraging disjoint-union structures for efficient state and epistemic class management. It extends the 'ATLIrModel' with additional capabilities tailored to complex ATL modeling scenarios.

8.2.2 Models

8.2.2.1 SimpleModel

The 'SimpleModel' class serves as the main structure for model representation in model checking scenarios. It encapsulates basic information and functionalities required for various types of models, particularly those used in temporal logic verification.

Constructor and Basic Properties:

- The constructor initializes the model with a specified number of agents.

- Key properties include the number of states, transitions, and agents, as well as structures for representing the model's graph, pre-images, epistemic classes, and states.

Model Manipulation Methods:

- `add_transition` and `add_epistemic_relation` allow for the dynamic expansion of the model's graph and the establishment of epistemic relations.

- `resize_to_state` ensures that the internal structures can accommodate a specified number of states.

- `epistemic_class_for_state` and `epistemic_class_for_state_and_coalition` provide mechanisms for querying epistemic classes.

Strategy and Transition Analysis:

- The class includes methods for analyzing possible strategies in individual states and sets of states, as well as for specific coalitions.

- `to_atl_perfect`, `to_atl_imperfect`, and similar methods enable conversion of the model to various formalisms like ATLIr, ATLir, and SLIr.

Subjective Semantics and Simulation:

- The `to_subjective` method adjusts the model for subjective semantics in ATLir, adding an initial state for a specific coalition.

- `simulate` provides a mechanism for interactive exploration of the model's behavior based on agent decisions.

Overall, the 'SimpleModel' class acts as a comprehensive framework for representing and manipulating models in temporal logic verification, offering extensive functionalities for epistemic relation handling, strategy analysis, and conversion to various logic formalisms.

8.2.2.2 GlobalModel

The ‘GlobalModel‘ class is designed to represent a global model, primarily used in the context of model checking. This class integrates local models and manages the reduction and representation of the global state space.

Constructor and Initial Setup:

- The constructor initializes the global model using local models, reductions, bounded variables, persistent states, coalitions, goals, logic type, formula, epistemic visibility, semantics, and initial conditions.
- It parses ATL and CTL formulas and determines the coalition based on the logic type.

Core Functionalities:

- `generate` and `generate_part` methods build the global model based on the provided parameters and local models.
- The model generation process considers both synchronous and asynchronous semantics.
- Epistemic relations are prepared using `_prepare_epistemic_relation`.
- Transition information is added to the model using `_add_index_to_transitions` and `_compute_shared_transitions`.

State and Transition Management:

- The class manages states and transitions, including enabling transitions, computing successors, and adding new states and transitions to the model.
- It utilizes a partial order reduction algorithm (`_iter_por`) for efficient state space exploration.

Utility Methods:

- Methods like `_successor`, `_add_state`, and `_copy_props_to_state` are used to manage state transitions and properties.

- The class also includes methods to compute enabled transitions and to check conditions for synchronous transitions.

Properties:

- Properties include the formula, model, local models, states count, transitions count, and model name.
- These properties provide access to essential aspects of the global model.

In summary, the 'GlobalModel' class offers a comprehensive structure for representing and managing a global model in model checking, encapsulating local models, handling state transitions, and integrating various semantics and epistemic relations.

8.3 Challenges and Lessons Learnt

Implementing the STV model-checker presented several significant challenges, each contributing valuable lessons to the development process. The primary focus was on how to effectively represent model structures in memory, balancing considerations of memory usage against the time complexity of essential operations during model generation and verification. This section highlights the key challenges encountered and the insights gained from addressing them.

Optimizing Memory Representation and Access Complexity The foremost challenge lay in representing model structures in a way that optimized memory usage while ensuring efficient access to necessary information. The main concern was the time complexity involved in operations such as verifying the uniqueness of newly generated states and retrieving lists of pre-states for a given state. This necessitated a careful design of data structures and algorithms to manage memory efficiently without compromising on the speed of access, crucial for large-scale model checking tasks.

Verification Process and State Management Another significant hurdle was related to the verification process itself, particularly in managing the dynamically growing set of winning and reachable states. This aspect of the tool required a robust and scalable approach to track and update state information as the verification process progressed, ensuring both accuracy and efficiency.

Maintaining Code Structure During Expansion As the STV model-checker evolved, incorporating new functionalities posed its own set of challenges. Maintaining a clean and structured codebase while expanding the tool's capabilities was crucial for long-term maintainability and ease of further development. This necessitated a disciplined approach to software design, emphasizing modularity and readability, to accommodate the tool's growth without compromising on code quality.

Lessons Learnt These challenges brought forth several lessons in software engineering, particularly in the realms of data structure optimization, algorithmic efficiency, and software design principles. The experience underscored the importance of a well-thought-out design strategy that accommodates future expansions and modifications, balancing the trade-offs between memory usage, processing speed, and code maintainability.

9

Experimental Evaluation

This chapter is dedicated to the experimental evaluation of various verification and state-space reduction methods discussed in the preceding chapters. The primary objective is to assess the effectiveness, efficiency, and practicality of these methods when applied to a range of models that have been detailed earlier in this paper. Through this empirical analysis, we aim to provide a comprehensive understanding of how these theoretical approaches perform for different scenarios and with varying model complexity.

The evaluation encompasses a series of experiments designed to rigorously test each verification and state-space reduction technique. We apply these methods to a diverse set of models, carefully chosen to represent a wide spectrum of possible use cases and challenges in model checking. This approach ensures that our findings are robust and generalize across various contexts, offering valuable insights into the strengths and limitations of each method.

In addition to assessing the accuracy of each technique, a significant focus is placed on evaluating performance metrics such as computational time and memory usage. These metrics are crucial for understanding the scalability and practical viability of the methods in handling complex, real-world models. By analyzing these performance aspects, we provide concrete data that support decision-making processes regarding the selection and application of verification and state-space reduction techniques in different model checking scenarios.

Lastly, this chapter synthesizes the experimental results to draw meaningful conclusions about the overall efficacy of the studied methods. We discuss the implications of these findings for practitioners and researchers in the field of model checking, offering

recommendations based on empirical evidence. The insights gained from this experimental evaluation are intended to guide future developments and applications in the domain of verification and state-space reduction.

The content of this chapter is based on the following papers: Jamroga and Kurpiewski (2023); Jamroga et al. (2019a, 2022a,b); Kurpiewski and Marmsoler (2019); Kurpiewski et al. (2019b, 2023).

9.1 Bridge Endplay

In this section, we delve into an in-depth analysis of the endplay scenarios in the bridge card game, as introduced in Section 7.1. Our objective is to apply a spectrum of verification methods to different versions of this model, thereby assessing the effectiveness of these approaches in capturing the nuances and strategic complexities inherent in the game.

To achieve a comprehensive understanding, we first outline the specific aspects and variations of the bridge endplay model that will be subjected to verification. This includes different game scenarios, strategies employed by the players, and the dynamic nature of the game's progress. Each model variant is designed to encapsulate key elements of bridge endplay, as well as it's variations.

Subsequently, we apply a range of verification methods to these models. The methods selected for this analysis represent a cross-section of the latest advancements in model checking and state-space analysis. By applying diverse methods, we aim to evaluate their relative strengths and limitations in the context of the intricate decision-making processes in bridge endplay.

For each method applied, we present detailed results, including the effectiveness in identifying winning strategies, the computational efficiency, and the scalability of the approach. We compare these results across different model variants to draw insights into which verification techniques are most suitable for analyzing specific aspects of bridge endplay.

Finally, the insights garnered from this analysis are synthesized to provide recommendations on the applicability of these verification methods in bridge endplay modeling. This section aims to not only enhance our understanding of the model but also to

(n,k)	#states	tgen	Lower approx.		Upper approx.		Match	Exact
			tverif	%true	tverif	%false		(tg+tv)
$(1,1)$	11	0.0007	0.00007	100%	0.00004	0%	100%	0.12
$(2,2)$	346	0.011	0.0008	100%	0.0003	0%	100%	2.42 h*
$(3,3)$	12953	0.73	0.07	85%	0.01	15%	100%	timeout
$(4,4)$	617897	35.19	348.37	90%	0.72	10%	100%	timeout
$(5,5)^\star$	2443467	132.00	8815.73	100%	4.216	0%	100%	timeout

Table 9.1: Experimental results: solving endplay in bridge by fixpoint approximation

contribute to the broader field of game theory modeling and analysis through empirical findings.

9.1.1 Standard Scenario

We start with the standard version of the bridge game from Section 7.1.1, where the goal is to win the most tricks. The formula we want to verify is $\varphi \equiv \langle\langle \mathbf{S} \rangle\rangle_{\mathrm{ir}} \mathbf{F}\mathsf{win}$, which means that the South player has a strategy to eventually win the game. As the verification method we use fixpoint approximations from Chapter 3. The results of the experiments are shown in Table 9.1. The columns present the following information:

- the parameters of the model (n,k),
- the size of the state space (#states),
- the generation time for models (tgen),
- the verification time (tverif) and the percentage of instances for which the output of approximate verification has been conclusive for the lower approximation tr_L (%true) and the upper approximation tr_U (%false);
- the percentage of cases where the bounds have matched (Match), and
- the total running time of the exact $\mathbf{ATL}_{\mathrm{ir}}$ model checking with MCMAS (tg+tv).

The times are given in seconds, except where indicated. We ran the experiments for up to 48h per instance. The results in each row are averaged over 20 randomly generated instances, except for $(\star)$ where only 1 hand-crafted instance was used. In case of the approximate model checking for random $(5,5)$ models, the program was not

(n,k)	#states	tgen	Lower approx.		Upper approx.		Match	Exact
			tverif	%true	tverif	%false		(tg+tv)
$(1,1)$	19	0.001	0.0001	100%	0.0001	0%	100%	9.68 h*
$(2,2)$	713	0.04	0.01	100%	0.004	0%	100%	timeout
$(3,3)$	52843	5.18	18.61	65%	0.58	15%	80%	timeout
$(4,4)$	memout							timeout

Table 9.2: Experimental results for absent-minded declarer by fixpoint approximation

(n,k)	#states	tgen	Lower approx.		Upper approx.		Match	Exact
			tverif	%true	tverif	%false		(tg+tv)
$(1,1)$	19	0.002	<0.0001	0%	<0.0001	0%	0%	14.93 h*
$(2,2)$	735	0.05	0.001	0%	0.004	10%	10%	timeout
$(3,3)$	60563	6.34	0.04	0%	0.58	35%	35%	timeout
$(4,4)$	memout							timeout

Table 9.3: Absent-minded declarer, approximation tr_{L2}

even able to complete model generation due to memout. For the exact model checking of random $(2,2)$ models, timeout was obtained in most instances.

9.1.2 Absent-Minded Declarer

We move to the absentminded version of the bridge game, as described in Section 7.1.3. The results of the experiments are shown in Table 9.2. Again we used fixed-point approximations as the verification method. Note that, for this class of models, the bounds do not match as tightly as before. Still, the approximation was conclusive in an overwhelming majority of instances. Moreover, it grossly outperformed the exact model checking which was (barely) possible only in the trivial case of $n = 1$.

The models are not turn-based, not lockstep, and not of perfect recall. Since they are not lockstep, approximations tr_{L2} and tr_{L3} do not have to coincide. In Table 9.3, we present the experimental results obtained with tr_{L2}, which show that the improved approximation tr_{L3} provides tighter lower bounds also from the practical point of view.

(n,k)	#states	tgen	Lower approx.		Upper approx.		Match	Exact
			tverif	%true	tverif	%false		(tg+tv)
$(1,1)$	11	<0.0001	<0.0001	100%	<0.0001	0%	100%	0.12
$(2,2)$	346	<0.0001	<0.0001	56%	<0.0001	44%	100%	2.42 h*
$(3,3)$	12953	0.06	<0.0001	62%	<0.0001	38%	100%	timeout
$(4,4)$	617897	4.64	0.56	62%	0.26	38%	100%	timeout
$(5,5)^\star$	2443467	34.00	3.0	100%	2.0	0%	100%	timeout
$(5,5)$	15190971	124.00	8.5	50%	6.0	50%	100%	timeout
$(6,6)^\star$	70094091	3779.00	667.0	100%	78.0	0%	100%	timeout

Table 9.4: Results of the optimized algorithm for the bridge model by fixpoint approximation

9.1.3 Optimized Algorithm

The experiments in Section 9.1.1 were conducted using a straightforward, one can even say naive, implementation of the fixpoint approximation algorithms. Already for that implementation, the results looked very promising, scaling up to millions of states. In this section, we show that one can develop and apply various optimization techniques, both domain-independent and domain-specific, that complement the general approximation scheme. We have focused on optimizing the data structures and operations on them. Most likely, other kinds of optimizations can be applied as well; we plan to study the issue in the future.

The optimized algorithms were implemented in C++ which offered better control of data management than Python. We ran the experiments in the same way and in the same environment as before. The results of the experiments with the optimized algorithms are shown in Table 9.4. When compared to Table 9.1, one can see further dramatic speedup and a lower memory usage. The algorithms were able to generate and verify a model with over 70 million states in less than 75 minutes. Perhaps more importantly, the verification of the handpicked $(5,5)^\star$ model (which marked the limit of our capability in Section 9.1.1) ran almost 3000 times (!) faster than with the straightforward implementation. This strongly suggests that the potential for further improvement is still large.

Our optimized algorithms reached their limit with randomly generated $(6,6)$ models,

(n,k)	AL	#states	Lower abstraction			Upper abstraction			Match
			tgen	tverif	%true	tgen	tverif	%false	
$(5,5)$	1	1328192	258	1.4	0%	330	1.0	0%	0%
$(5,5)$	2	2481220	323	2.8	0%	412	3.4	0%	0%
$(5,5)$	3	6223840	420	5.4	0%	500	7.6	0%	0%
$(5,5)$	4	9124109	490	8.25	50%	612	9.25	25%	75%
$(5,5)$	5	15190971	124	8.5	50%	124	6.0	50%	100%

Table 9.5: Verification with abstraction: results for the bridge model may/must abstraction by fixpoint approximation

(n,k)	AL	#states	Lower abstraction			Upper abstraction			Match
			tgen	tverif	%true	tgen	tverif	%false	
$(6,6)^\star$	1	272113	680	<1	0%	1203	<1	0%	0%
$(6,6)^\star$	2	408127	714	1	0%	989	1	0%	0%
$(6,6)^\star$	3	1420924	739	2	0%	993	2	0%	0%
$(6,6)^\star$	4	6925594	874	11	100%	1064	9	0%	100%
$(6,6)^\star$	5	13977070	1126	25	100%	1371	21	0%	100%
$(6,6)^\star$	6	70094091	3779	667	100%	3779	78	0%	100%

Table 9.6: Further results for abstractions of the bridge model may/must abstraction by fixpoint approximation

facing memout during model generation. A possible way to overcome the limitation is by state abstraction of the model.

9.1.4 Abstraction

We have conducted three series of experiments with abstractions of the bridge endplay models. First, we re-approached the largest models that we managed to verify in the previous sections. We looked at the balance between accuracy and performance, obtained by different granularity levels of abstraction. The results are shown in Tables 9.5 and 9.6. Table 9.5 presents the output of experiments for randomly generated models $(5,5)$, i.e., models of full play with each player holding initially 5 cards. We studied all possible levels of abstraction from $r = 1$ (all ranks of cards removed) up. The results for $r = 5$ (full model, no abstraction) form the baseline; we repeat them after Section 9.1.3.

(n,k)	AL	#states	Lower abstraction			Upper abstraction			Match
			tgen	tverif	%true	tgen	tverif	%false	
$(13,1)$	6	11	<1	<1	65%	<1	<1	15%	80%
$(13,2)$	6	243	<1	<1	45%	<1	<1	20%	65%
$(13,3)$	6	12504	<1	0.1	55%	<1	<1	15%	70%
$(13,4)$	6	545323	10	0.5	50%	12	0.45	30%	80%
$(13,5)$	6	6406684	562	7.2	20%	646	9.2	0%	20%

Table 9.7: Further results for abstractions of the bridge model may/must abstraction by fixpoint approximation

Table 9.6 presents an analogous study for the handpicked model $(6,6)$, the same as in Section 9.1.3.

Finally, we model checked existence of winning strategies in k-endplay with the full deck of cards ($n = 13$), for a fixed level of abstraction ($r = 6$) and different values of k. The results are shown in Table 9.7. This series of experiments is especially interesting, as it best captures what happens in reality. Middle-level human players deal with the complexity of the full deck of cards by discerning between ranks from A down to 10, and abstracting away from the "low ranks" between 9 and 2.

In all the tables, the columns present the following information:

- the parameters of the model (n,k), i.e., the number of cards per suit in the deck (n) and the initial number of cards per player (k);
- the level of the abstraction (AL$= r$);
- the size of the state space after abstraction (#states);
- the generation time for the lower and the upper abstraction of the model (tgen),
- the time and the output of verification (tverif, %true, %false) for model checking of the lower and the upper abstraction,
- the percentage of cases where the bounds have matched (Match).

The times are given in seconds. The experiments were run in exactly the same environment as before. We used the optimized version of our fixpoint algorithm, presented in Section 3.2. The results in each row are averaged over 20 randomly generated instances, except for $(\star)$ where only 1 hand-crafted instance was used.

Conf.	DominoDFS	MCMAS	Approx.	Approx. opt.
$(1,1)$	0.0006	0.12	0.0008	< 0.0001
$(2,2)$	0.01	8712*	0.01	< 0.0001
$(3,3)$	0.8	timeout	0.8	0.06
$(4,4)$	160	timeout	384	5.5
$(5,5)$*	1373	timeout	8951	39
$(5,5)$	memout	timeout	memout	138
$(6,6)$*	memout	timeout	memout	4524

Table 9.8: Results for Bridge by domination-based DFS

9.1.5 Domino DFS

For the next set of experiments we use Domino DFS algorithms from Section 4.2.4 and the model of the standard version of the bridge game endplay. Table 9.8 collects the experimental results for models of the Bridge scenario. The first column identifies a subclass of the models. For each such subclass, we have run tests on 50 randomly generated card deals, except for the configurations marked with (*) where we were only able to run tests on a single handcrafted instance of the model due to timeout or memout. The other columns present the average performance of model checking (model generation + verification time): first for our new algorithm (DominoDFS), and then for the reference tools. We provide a comparison to MCMAS and the fixed-point approximations. The approximations are used in two variants: the basic one (Approx.) from Section 3.1 and the optimized one (Approx. opt.) from Section 3.2.

9.1.6 Discussion of Results

Fixpoint Approximation

In the experiments, our approximations offered a dramatic speedup. The exact model checking of φ was infeasible except for the simplest models (hundreds of states), even with an optimized symbolic model checker like MCMAS. In contrast, our bounds were verified for models up to millions of states. Moreover, our approximations obtained an astonishing level of accuracy: the bounds matched in 100% of the analyzed instances, thus producing fully conclusive output.

Abstraction

The first two series of experiments show, unsurprisingly, that stronger abstraction saves much of the verification time, but also removes from the model information that can be vital for the verification output. In particular, too strong abstraction removes information needed to identify more sophisticated winning strategies. Of course, when combining *two* approximation techniques, one should expect the gap between the bounds to be significant. Still, the gap in our experiments was clearly due to abstraction, and not because of the fixpoint approximation (this is evident from the baseline results). On a more positive note, the results in Table 9.6 demonstrate that, for some models, conclusive model checking with abstraction can be done faster by an order of magnitude, compared to model checking without abstraction. Note also that conclusive verification for the $(6,6)^\star$ model was possible without discerning half of the card ranks in the deck (abstraction level $r = 4$).

The results in the last series are especially hopeful. With abstraction, we were able to verify models which had been beyond our reach in their full form. The accuracy of the approximation ranged between 20% and 80%. This means that combining approximation on formulae and approximation on models allows for model checking of some instances that are too complex for either of the approximation methods alone.

Domino DFS

The results show that DominoDFS significantly outperforms MCMAS. It also successfully competes with the basic implementation of fixpoint approximation. We also note that this approach can handle models that do not submit to the fixpoint approximation scheme. This allows us to suggest the following mete-procedure for verification of enforceability in models with incomplete information: first try optimized fixed-point approximations, if this fails then apply DominoDFS, finally try your luck with remaining tools.

Somewhat surprisingly, none of the heuristics have performed notably better or worse than the simple reduction, hence we omit their performance from the tables.

9.2 Castles Benchmark

In this section we focus on the Castles model from Section 7.9.1. We use this benchmark to evaluate the performance of the dominance-based DFS strategy synthesis algorithm.

Conf.	DominoDFS	MCMAS	SMC
$(1, 1, 1)$	0.3	65	63
$(2, 1, 1)$	1.5	12898	184
$(3, 1, 1)$	25	timeout	6731
$(2, 2, 1)$	25	timeout	4923
$(2, 2, 2)$	160	timeout	timeout
$(3, 2, 2)$	2688	timeout	timeout
$(3, 3, 2)$	timeout	timeout	timeout

Table 9.9: Results for Castles benchmark by domination-based DFS

As the Castles model does not satisfy the necessary conditions for fixpoint approximations from Chapter 3, we do not evaluate this method here.

We verify the formula $\varphi_{\text{Castles}} \equiv \langle\langle c12 \rangle\rangle F \mathit{castle3Defeated}$, i.e., whether the coalition of the workers from castles 1 and 2 can defeat the third castle.

9.2.1 Domino DFS

Table 9.9 collects the experimental results for Castles. Each triple in the Configuration column refers to the number of workers assigned to the corresponding castles. The initial amount of health points for each castle is 3. Since the benchmark does not fulfil the necessary condition for fixpoint approximation from Section 3.1, we only compare the performance of the dominance-based synthesis to MCMAS and SMC. The times are given in seconds, and the timeout is 4 hours.

9.2.2 Discussion of Results

The results show that Domino DFS algorithm outperformed both MCMAS and SMC. It is also important to note that this method can be applied to scenarios for which the fixpoint approximations approach does not end with conclusive results.

9.3 Drones Benchmark

In this section we focus on the Drone model from Section 7.2.

9.3.1 One-Criterial Strategy Optimization

We evaluate the algorithm from Section 5.2 on models of a team of drones equipped with an initial number of energy points, inspecting a fixed, two-dimensional map. Each action of a drone consumes an energy point. Once the energy level reaches zero, the drone becomes useless. The limited precision of the drone's telemetry is modeled by making selected locations indistinguishable. The drone can attempt to fly in one of the four directions (NESW) unless there is an obstacle. It can also wait, staying in the current place. In some locations, the outcome of its actions can be influenced by the wind that alters the direction of the movement. The states that contain those locations are called *non-controlled*. The action of waiting is thus active: a drone stays in its current location, opposing the wind. It can also decide to drift with the wind, executing the action Fly.

We consider two subclasses of the drone benchmark: one with a single drone, and one with two drones inspecting the same map. The model is scalable w.r.t. the number of initial energy points. The basic strategy to be optimized is always the same: drift with the wind (if detected) or wait, being passively moved by the environment until the battery runs out. The characteristics of the basic strategy is presented in Fig. 9.2. The boxed line shows the number of *reachable states*, i.e., the states that must be taken into account when the strategy is executed (including the states reachable by epistemic indistinguishability links). The diamond line indicates the number of *reachable uncontrollable states*, i.e., reachable states where the wind is present.

The experimental results for the single-drone model scaled w.r.t. the energy points of the drone are presented in Fig. 9.1. As it can be observed, the simple approach allows for substantial reductions of the basic strategy, leading to 50%–75% smaller sets of outcome states.

The final batch of experiments was designed to further inspect the efficiency of our approach. To this end, we used the two-drone variant of the Drone benchmark, and randomly transformed approximately 50% of locations by adding non-controllable actions (i.e., the wind). The output of the experiments is presented in Table 9.10, with the percentages showing the fraction of reachable states that are left after the optimization. The results consistently display a high degree of reduction: the optimized strategies have

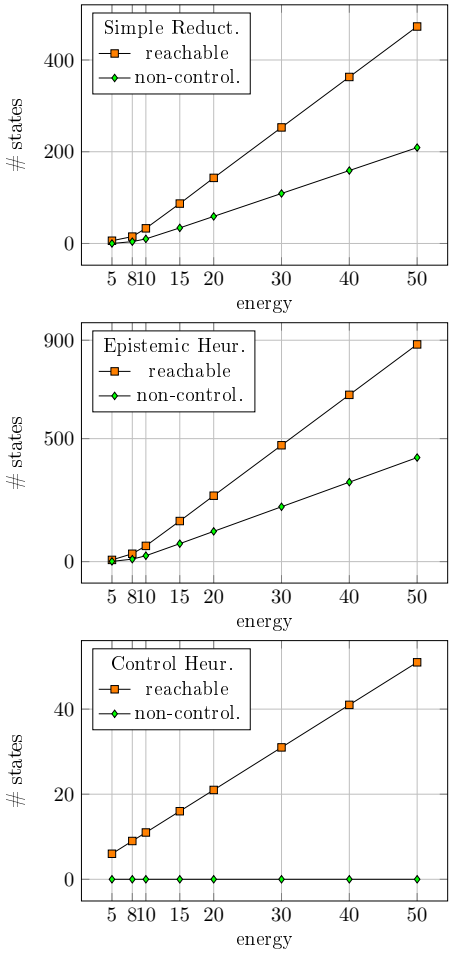

Figure 9.1: Reduced Strategies (Single Drone) by domination-based optimization

roughly 100 times smaller sets of reachable states (as well as non-controllable reachable states) regardless of the heuristics being used.

9.3.2 Multi-Criterial Strategy Optimization

The output of the experiments is presented in Table 9.11 All running times are given in seconds. The timeout was set to 90 seconds. In case of strategy optimization, this was split into two parts: 30 seconds for the strategy generation, and 60 second for its optimization.

The first columns present information about the model configuration, its size and generation time. The next seven columns describe the output of our algorithms, i.e.,

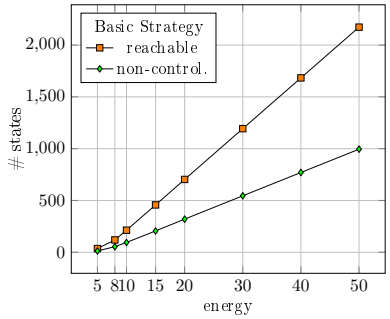

Figure 9.2: Basic Strategy (Single Drone) by domination-based optimization

Basic strat.		Simple Reduct.		Epistemic Heur.		Control Heur.	
reach	nctr	%reach	%nctr	%reach	%nctr	%reach	%nctr
782	256	1	2	1	2	5	8
1022	350	4	6	1	2	1	2
1147	385	2	4	5	6	3	4
1257	430	10	13	9	13	5	9
1601	512	3	5	3	6	2	4
1853	625	4	5	4	4	4	4
2527	834	1	2	1	2	1	2

Table 9.10: Randomized Example (2 Drones / 5 Energy Pts) by domination-based optimization

the randomly generated strategy with perfect information and its optimized version. The last part of the tables contains the reference results from the algorithms used for comparison: lower and upper Approximation and Domino DFS method.

The table headers should be interpreted as follows:

- *#st*: number of states in the model
- *G.time*: generation time for the model/strategy

			Strategy Perfect Info			Simplified Strategy				Approximation		Domino DFS	
Map	#st	G. time	G. time	#st	#ep	G. time	#st	#ep	%ir	Time	Conclusive	Time	True
5	330	0.078	0.043	38	13	36.003	13	1	60%	0.036	0%	9.012	90%
10	10648	3.420	1.284	74	33	42.478	30	5	60%	1.895	0%	90	TIMEOUT

Table 9.11: Drone Model results by domination-based optimization, fixpoint approximation and domination-based DFS

- $\#str$: number of states reachable in the strategy
- $\#ep$: number of states in which the strategy uniformity was broken
- $\%ir$: percentage of cases in which optimized strategy was a uniform strategy
- $Time$: time used by the Approximation/Domino DFS algorithm
- *Conclusive*: percentage of cases in which the result of fixpoint approximation was conclusive
- $True$: percentage of cases in which Domino DFS returned a winning strategy, timeout was reached in other cases.

As the results show, our method performed very well, especially for models with relatively small information sets. This is even more evident in comparison to the reference algorithms. The Domino DFS method ended mostly with timeout for larger models, and the fixpoint approximations gave mostly inconclusive results. In contrast, our optimized strategies obtained pretty good elimination of conflicts, and in many cases produced ideal, i.e., fully uniform strategies.

The results also show clearly that our optimization algorithm works best in situations when the size of the epistemic classes is relatively small. For the logarithmic size of the epistemic classes, the optimized strategy was always a uniform strategy (!). As for the setting with the linear size, the optimization-based algorithm was not as good, but still gave a reduction of conflicts of about 40%. Even in that case, it produced ideal strategies in $10 - 20\%$ of instances. It is also worth pointing out that, for the Drone benchmark, our optimization returned uniform strategy in about 60% cases.

Finally, we noted that our algorithm is an anytime algorithm, which means that optimized strategy will always be returned, irrelevant of the given timeout. It is especially important when many tests must be executed in the short period of time.

9.3.3 Multi-Criterial Strategy Optimization for Coalitions

The output of our experimental evaluation for synthesis of coalitional strategies is presented in Table 9.12. For the experiments, the Drone benchmark was selected with coalition of of two drone agents. As the results show, our algorithm obtained a high level of optimization of the initial, perfect information, strategy. Most importantly, the

			Strategy Perfect Info			Simplified Strategy			
Map	#st	G. time	G. time	#st	#ep	G. time	#st	#ep	%ir
3	667	0.85	0.397	35	11	12.031	9	1	60%
5	31122	69.265	107.428	587	728	60.8	87	58	40%

Table 9.12: Drone model results for coalitions by domination-based optimization

procedure produced ideal strategies in 60 and 40% of the instances, respectively, thus providing a conclusive answer to the model checking question in about half of the cases.

9.3.4 Discussion of Results

As the experiments show, the result of the formula depends mostly on the initial energy of the drones. If given enough energy, drones can visit every place on the map, hence detecting any pollution. On the other hand, even a drone with very small capacity of the battery can detect something.

9.4 Machines and Robots

In this section we apply our verification methods to the factory model from Section 7.3.

9.4.1 Formulae

We considered four different properties that are important for the safe functioning of the presented factory scenario. Formulas representing these properties are as follows:

- $\varphi_1 : \langle\langle R \rangle\rangle \mathbf{F}(produce_n)$
- $\varphi_2 : \langle\langle R \rangle\rangle \mathbf{F}(\neg stuck \wedge produce_n)$
- $\varphi_3 : \langle\langle R \rangle\rangle \mathbf{F}(energy > 0 \wedge produce_n)$
- $\varphi_4 : \neg\langle\langle R' \rangle\rangle \mathbf{G}(\neg produce_n)$

where:

- $produce_n$ denotes that each machine has produced at least n items, and

En.	# Ch.	# Stor.	Pr. t.	It. l.	# St.	Gen.
inf	0	0	0	1	3581	0.76
inf	0	0	1	1	4161	0.83
inf	0	0	0	2	13912	3.07
10	0	0	0	1	28667	6.23
10	1	0	0	1	48426	10.65
inf	0	1	0	1	4670	1.32

Table 9.13: Generation results for the machines and robots model

- *stuck* denotes that a machine is stuck, i.e., that its input requirements are met, but its output isn't empty.

The first three of the presented formulas describe the properly functioning of the factory. φ_1 means that when robots work together they can make sure that each machine will produce the required number of items. As one can see this is a rather basic property, but it's a necessary condition for the properly functioning of the factory. Still this may be not enough to describe an efficiently operating factory. That we try to achieve using formula φ_2, where robots not only need to make each machine produce the required number of items, but they also need to make sure that each machine will produce a new item when all required items are delivered. This way we know that a factory setting is efficient, i.e., there is no redundant waiting time for machines. The third formula φ_3 describes configurations in which no robot will be left with zero energy after executing its commands.

The last formula φ_4, unlike the others, describes a security property of the model. Imagine a possible scenario: someone wants to disturb the work in the factory by hijacking, or maybe just disabling some of the robots. The question is, how secure is the factory against such an attack? That is what is the minimal subset of the robots that the adversary must take control of, in order to disturb the factory production line?

9.4.2 Factory Configurations

Different variants of the model may result in different number of states. For example adding storage areas to the factory layout changes the actual size of the model and may have an impact on the verification times. In order to have a good understanding of

conf.	#states	IR t.	IR	iR t.	iR (appr.)
(1, 0)	3581	1.1	true	0.3	true
(3, 0)	26039	2.4	true	1.1	true
(3, A: 5)	72573	8.3	true	5.7	true

Table 9.14: Results for model checking φ_1 by fixpoint approximation

how such changes may affect the experiments, we present some details regarding the generation process of these different models in Table 9.13. Headers should be interpreted as follows:

- En. - initial energy of the robots; inf means that robots don't use energy
- # Ch. - number of charging stations in the factory
- # Stor. - number of storage areas in the factory
- Pr. t. - production time for the machines
- It. l. - item limit for the machines, i.e. how many items can any machine produce
- # St. - number of states in the generated model
- Gen. - generation time of the model in seconds

As the results show, the modification that drastically affects the size of the model is adding energy to the robots. For example lets take a look on the first, simplest configuration: no energy limit, no charging stations and storage areas, no time requirements for productions and production limited to one item per machine. Such a simple model consist of 3581 different states. If we change this configuration by only specifying initial energy for robots of 10 units, the number of states in the generated model goes up to 28667 (this corresponds to an increase in size by a factor of 8). If we also add one charging station to the model, thus allowing robots to recharge their energy, we end up with 48426 states, which is 13 times more than in the initial, simplest configuration.

9.4.3 Basic Production

In this section we consider formula $\varphi_1 : \langle\langle R \rangle\rangle \mathbf{F}(produce_n)$. The goal for the coalition of robots is to reach a state, in which each machine has produced at least n items. We will test this property in different configurations of the model. First, we begin with a classic model based on a factory layout as presented in Fig. 7.9. The model consist of two robots and two machines. Robots can move freely, apart from places marked as an obstacle. Robots don't consume energy and can carry at most one item - there is no storage in the factory. Requirements for machines are simple: the first machine doesn't need anything to produce an item and the second machine needs one item from the first machine. By manipulating the parameters of the model we can change: items limit, machines requirements and production times for each machine. We can also modify parameter n of the formula. As the experiments show, the formula always holds, both under perfect and imperfect information, except when there is a conflict in the parameters. A conflict occurs when, for example, machine requirements are constructed in such a way, that at least one of the machines will be always blocked. Results are shown in Table 9.14. For simplicity, machines are named A and B. The configuration should be interpreted as follows: the first parameter defines the limit for the production, and the second one defines production times for machines. For simplicity, if production times for both machines are 0, we will write only 0 as the second parameter. If a machine requires some time to produce an item, we will write *machine* : *number*, where machine is the letter representing the machine and number is production time for this machine. If some machines are omitted we assume that production time for them is by default 0.

The results should be interpreted as follows:

- conf. - configuration, as explained above
- # states - number of states in the generated model
- IR t. - perfect information verification time in seconds
- IR - result under perfect information
- iR t. -imperfect information verification time in seconds
- iR (appr.) - approximated result for the imperfect information

conf.	#states	IR t.	IR	iR t.	iR (appr.)
(1, 0)	3581	0.92	true	0.18	true
(3, 0)	26039	0.002	false	0.002	false
(3, A: 5)	72573	0.005	false	0.005	false
(3, A: 5, st: 1)	198762	10.5	true	8.7	true

Table 9.15: Results for model checking φ_2 by fixpoint approximation

Of course, in such a simple environment, results of the experiments are as expected. Perhaps one can get more interesting scenarios by specifying finite charges for the robots. In this version of the model, apart from the previous parameters, we can also modify the number of charging stations, their positions and initial charge of the robots. Again, we conducted some experiments, using previous configurations with additional parameters. First we begun with a simpler scenario, i.e., a factory without charging stations. In such an environment one can already suspect the result of the formula. The strategy for robots seems to be simple: find shortest path to fulfill machine requirements. Of course, there are some obstacles along the way, such as avoiding collisions.

While conducting experiments it is possible to find the minimal initial charge for the robots in a given configuration, under which the formula is satisfied. There can be some differences based on the considered type of information. Under imperfect information, robots may require more energy than under perfect information. We can also manipulate positions of the charging station to find optimal positioning.

9.4.4 No Stuck Time

As said before, proper functioning of the factory is sometimes not enough. Sometimes we need to ensure that a considered configuration is not only stable, but also efficient. After the previous section, we know under which configurations the system is stable, i.e. robots can enforce production of n items from each machine. Now, we want to know when the system is efficient, i.e. each machine can produce a new item immediately after its requirements are fulfilled (when it receive all items needed for production from the robots). In other words, we will call a configuration efficient, if a coalition of robots has a strategy to ensure that each machine will produce the required number of items and while doing so, no machine will be stuck. Our stuck property in the model is

conf.	#states	IR t.	IR	iR t.	iR (appr.)
(1, 0, en: 5, ch: 0)	2681	0.0002	false	0.0001	false
(1, 0, en: 9, ch: 0)	20439	0.6	true	0.05	true
(1, 0, en: 5, ch: 1)	5660	0.0004	false	0.0003	false
(1, 0, en: 6, ch: 1)	16489	1.3	true	0.3	true

Table 9.16: Results for model checking φ_3 by fixpoint approximation

sticky - once a machine is stuck it is stuck forever. We took the configurations from the previous section and run our model-checking algorithm again, this time with formula φ_2. Results are shown in Table 9.15.

9.4.5 No Robot Left Behind

In order to talk about energy level we need a model with energy. When checking formula φ_3 we will only consider models with energy. Based on its initial level, robots may or may not be able to fulfill their duties. When thinking about the property described by formula φ_3, some questions arise: What is the minimal level of the initial energy required for each robot? What is the optimal positioning of the charging stations and how many are required? Does storage affect this property? How does imperfect information affect this property? How do production times of the machines affect this property? We try to answer these questions by conducting several experiments on different configurations. First, lets describe basic requirements for our scenario. As previously, we consider a 6×6 factory layout with two machines and two robots. We will consider two configurations of the production times for the machines: [A:0, B:0] and [A:2, B:3]. This way, we can see how production times will affect our results. As for the number of items we want to produce, we choose 1 and 2. As shown before, the number of states grows fast, when we add energy to the robots, so it is only sensible to limit the size of the model by using low limits for production. When combined together, our requirements give us four different configurations and we try to find optimal configurations for the rest of the parameters: initial energy, charging stations and storage areas. The results are summarized in Table 9.16.

#robots	normal	charging	storage
2	1	1	1
3	2	1	2

Table 9.17: Results for model checking φ_4 by fixpoint approximation

9.4.6 Security

There are various ways to describe security properties. In our factory scenario, the level of security can be described by the minimal number of robots that an adversary needs to take control of, in order to be able to disturb the work of the factory. For example, lets say that we have a factory with 10 robots. Rendering 3 of them malfunctioning won't affect the system, but if the adversary takes control of the fourth robot, then the rest of them won't be able to ensure proper functioning of the factory anymore. In this situation the minimal number of robots needed to disturb the production is 4 out of 10. Of course, the result may vary depending on the factory configuration, layout, and even on the concrete robots that are hacked. Work of some of the robots may be more important to ensure proper production, than the work of other robots. Table 9.17 show results of the experiments conducted on a smaller, 4x4 factory. First column (#robots) contains number of robots in the tested configuration. Next columns shows minimum number of robots that need to be hacked in order to disrupt the work of the factory in the different variants of the model: without energy and storage, with energy and charging stations, with storage areas.

9.4.7 Discussion of Results

In our research, we employed strategic model checking to ascertain the minimal initial energy level required for robots to successfully achieve their designated goals. Furthermore, this approach was utilized to determine the minimum number of robots necessary to maintain operational efficiency within a factory environment, thereby ensuring the fulfillment of production objectives. Our findings offer valuable insights into the security aspects of the production system, revealing the least number of robots that must be compromised to disrupt production processes significantly.

		Strategy Perfect Info			Simplified Strategy				Approximation		Domino DFS	
#st	G. time	G. time	#st	#ep	G. time	#st	#ep	%ir	Time	Conclusive	Time	True
10	0.014	0.031	10	5	42.033	7	0	100%	0.018	50%	0.57	100%
100	0.176	0.546	92	61	60.210	83	0	100%	0.519	20%	90	TIMEOUT
1000	9.401	22.001	882	629	61.865	780	0	100%	3.136	0%	90	TIMEOUT

Table 9.18: Random Model results with logarithmic epistemic classes by domination-based optimization, fixpoint approximation and domination-based DFS

		Strategy Perfect Info			Simplified Strategy				Approximation		Domino DFS	
#st	G. time	G. time	#st	#ep	G. time	#st	#ep	%ir	Time	Conclusive	Time	True
10	0.009	0.023	10	5	24.048	6	3	20%	0.017	80%	1.14	100%
100	0.202	0.489	94	58	54.253	66	36	10%	0.197	0%	90	TIMEOUT
1000	10.817	25.239	917	584	61.496	614	347	10%	2.647	0%	90	TIMEOUT

Table 9.19: Random Model results with linear epistemic classes by domination-based optimization, fixpoint approximation and domination-based DFS

During our experimental evaluations, we managed to verify strategic properties in models incorporating up to four robots and two machines. Despite the apparent limitations imposed by memory outages due to state space explosion, the models under consideration contained hundreds of thousands of states. Although the scale of our experiments, limited to a few robots and machines, might seem modest, the results yielded interesting insights into the operational dynamics of the system and its inherent capabilities.

9.5 Random Models

In this section we evaluate our algorithms on the random models from Section 7.8.

9.5.1 Multi-Criterial Strategy Optimization

The output of the experiments is presented in Tables 9.18 and 9.19. All running times are given in seconds. The timeout was set to 90 seconds. In case of strategy optimization, this was split into two parts: 30 seconds for the strategy generation, and 60 second for its optimization.

The first columns present information about the model configuration, its size and generation time. The next seven columns describe the output of our algorithms, i.e., the randomly generated strategy with perfect information and its optimized version. The last part of the tables contains the reference results from the algorithms used for comparison: lower and upper Approximation and Domino DFS method.

The table headers should be interpreted as follows:

- $\#st$: number of states in the model
- $G.time$: generation time for the model/strategy
- $\#str$: number of states reachable in the strategy
- $\#ep$: number of states in which the strategy uniformity was broken
- $\%ir$: percentage of cases in which optimized strategy was a uniform strategy
- $Time$: time used by the Approximation/Domino DFS algorithm
- $Conclusive$: percentage of cases in which the result of fixpoint approximation was conclusive
- $True$: percentage of cases in which Domino DFS returned a winning strategy, timeout was reached in other cases.

9.5.2 Discussion of Results

The results of our study indicate that our method exhibits superior performance, particularly in scenarios involving models with relatively small information sets. This performance advantage becomes even more pronounced when contrasted with reference algorithms. Notably, the Domino DFS method frequently resulted in timeouts for larger models, while the fixpoint approximation methods predominantly yielded inconclusive outcomes. In stark contrast, our optimized strategies demonstrated significant conflict elimination efficiency, often generating ideal, i.e., fully uniform strategies.

Furthermore, the empirical evidence unequivocally suggests that our optimization algorithm excels in contexts where epistemic class sizes are relatively modest. Remarkably, for epistemic classes of logarithmic size, our optimized strategy consistently

achieved a uniform outcome. Conversely, in scenarios characterized by linear-sized epistemic classes, although the optimization-based algorithm did not perform as optimally, it still managed to achieve a conflict reduction rate of approximately 40%. Notably, in these instances, it generated ideal strategies in 10-20% of the cases.

Crucially, our algorithm is designed as an anytime algorithm, guaranteeing the return of an optimized strategy regardless of the specified timeout. This feature is particularly advantageous for conducting a multitude of tests within a constrained timeframe, underscoring the algorithm's practical utility and efficiency.

9.6 Simple Voting

In this section we verify the Simple Voting model from Section 7.5. We conducted two sets of experiments to evaluate the performance of the approximate verification approach using local models from Section 3.3. The first set of experiments focused on the standard Asynchronous Simple Voting model, while the second set examined an extended version of this model that incorporated a revoting mechanism. In both cases, we compared the verification time required for fixpoint approximation on our locally approximated models against that of the standard fixpoint approximation applied to the full global model, as implemented in the STV tool. The results, which are discussed in detail below, demonstrate remarkable performance gains. It is important to note, however, that the process of approximate verification necessitates a preliminary step: the generation of the local approximating model itself.

9.6.1 Generating Approximated Models

The naive approach for local model synthesis would be to first construct the complete global model and subsequently project it onto the states and transitions specific to a given agent a. To circumvent the computational intractability of this method, we propose a more refined procedure. This approach translates the existence of a transition sequence between two global locations l and l' into a Computation Tree Logic (CTL) verification problem, which is then resolved by a dedicated model checker.

In our experimental evaluation, we employed the UPPAAL model checker Behrmann et al. (2004) as a sub-routine to generate an approximated local model. This model was subsequently used as input for the STV model checker to verify strategic abilities.

All invocations of UPPAAL were orchestrated by an auxiliary script. This script parses the model specification (provided in the .xta format with event-labeled edges) and constructs a corresponding model-query pair in a breadth-first search (BFS) manner, commencing from the initial local state.

For each candidate triplet (s, α, s'), composed of a source local state s, an event label α, and a target local state s', the script performs a systematic transformation. In our framework, every event available to the agent under study is associated with an auxiliary Boolean variable. This variable evaluates to true exclusively when its corresponding event was the last to occur at the current global state. For each source local state, every outgoing edge bearing the specified event label is duplicated. The duplicate is augmented with a guard condition reflecting the valuation of the source local state and an update statement that aligns with the last-occurred event. Fragments of the model outside the set of potential predecessors are truncated, while the remaining edges are appended with an opposing update assignment. To enforce a progressive semantics, all locations are designated as committed. The resulting model is then queried for the reachability of the target local state's counterpart, constrained by the condition that the auxiliary variable for the last-occurred event matches the one specified in the triplet.

Following the generation of candidate transitions, the synthesized local states are verified for livelocks. A livelock is defined as a cycle of global states and events in which the local state of the agent under study remains unchanged, and no events from its repertoire are involved. This verification is performed using the original global model specification.

The direct application of UPPAAL for local model generation proved to be inefficient. Specifically, when a query corresponding to a triplet (s, α, s') has no valid counterpart in the global model, the verifier is forced to explore the entire state space to determine its non-existence. To mitigate this, we implemented several key optimizations. First, we performed a semantic analysis of the homogeneous voter modules, revealing a crucial symmetry: no voter interacts transitively with another, as their sole interaction is with the coercer. This insight allowed us to conclude that the order of actions among different voters does not influence the composition of their local models. Consequently, we leveraged UPPAAL's process priority mechanism by assigning the highest priority to the voter under investigation. Furthermore, to expedite query termination, we appended

#V	Model generation			Verification		
	Global	Approx. Standard	Approx. Optimized	Global	Approx.	Result
2	0.04	6.60	6.54	<0.01	<0.01	TRUE
3	0.10	6.62	6.60	0.29	<0.01	TRUE
4	1.22	6.93	6.91	30.15	<0.01	TRUE
5	35.80	8.71	8.70	2659	<0.01	TRUE
6	1206	36.95	29.42	timeout	<0.01	TRUE
7	timeout	282.48	280.62	——	<0.01	TRUE
8	timeout	5539	4046	——	<0.01	TRUE
9	timeout			——	——	——

Table 9.20: Results for Asynchronous Simple Voting, approximation by local models

#V	Model generation			Verification		
	Global	Approx. Standard	Approx. Optimized	Global	Approx.	Result
2	0.82	19.43	19.27	8.20	<0.01	TRUE
3	131.61	26.44	19.28	timeout	<0.01	TRUE
4	timeout	524.93	19.25	——	<0.01	TRUE
5	timeout		19.34	——	<0.01	TRUE
6	timeout		19.40	——	<0.01	TRUE
7	timeout		19.41	——	<0.01	TRUE
8	timeout		19.43	——	<0.01	TRUE
9	timeout		19.44	——	<0.01	TRUE

Table 9.21: Results for Asynchronous Simple Voting with Revoting, approximation by local models

an edge from the target local state to an auxiliary sink location, which contains a single self-loop.

9.6.2 Results

The results of the experiments are summarized in Tables 9.20 and 9.21. The first column indicates the number of voters considered. In all scenarios, there were two candidates and one coercer. The tables present the model generation times for the standard global model and two approximated models: one without optimizations (standard) and one with optimizations (optimized) in UPPAAL. Verification was performed using a fixpoint algorithm, and the verified formula was $\varphi_1 = \langle\langle Voter_1 \rangle\rangle \mathbf{F}(vote_{1,1} \wedge \neg give_1)$, meaning that voter 1 has a strategy to eventually vote for candidate 1 without giving proof of her vote to the coercer. The approximated model was generated for agent $Voter_1$. All times are reported in seconds, with a timeout set to two hours.

All experiments were conducted on a machine equipped with a 3.0 GHz 8-core AMD Ryzen 7 5700X3D CPU and 64 GB of RAM. The model generation and verification times

were measured using the `time` command in a Linux environment to ensure precise and consistent timing measurements.

9.6.3 Discussion of Results

As the experimental results demonstrate, the model generation times for a small number of voters are lower for the global model compared to its approximate counterparts. This trend, however, reverses as the voter population scales. The optimized version of the approximate model consistently outperforms the standard version in terms of generation speed. Nevertheless, its time complexity still exhibits exponential growth in the scenario without revoting. In stark contrast, when revoting is incorporated, the optimized approach demonstrates linear time complexity, enabling the generation of a model for 50 voters in under one minute.

A significant disparity is also observed in verification times: the global model's verification time increases exponentially, while that of the approximate model remains constant. This constant verification time is a direct consequence of the invariant size of the approximated model, as the local model of each voter is independent of the total number of voters in the system.

Notably, the optimization yields a more substantial reduction in model generation time for the revoting scenario. In the absence of revoting, the exponential time complexity is primarily attributable to the detection of potential livelocks. When no witnessing livelock cycle is present, the model checker is compelled to generate the entire fragment of predecessor states for the source local state before reaching a conclusion.

It is worth noting that both the standard and optimized versions present an opportunity for further efficiency gains through the parallel computation of reachable successors from the already discovered states of the local model. Such a parallelization strategy would facilitate more efficient generation of the approximated model, particularly for systems with a larger number of voters.

9.7 SELENE

In this section we present the experimental results for the SELENE case study introduced in Section 7.6.

$$\Phi_1 \equiv \langle\langle Coercer \rangle\rangle \mathbf{F} \left(\mathsf{finished} \wedge \left(\bigwedge_{v \in A} \neg \mathsf{vote}_{v,1} \rightarrow \mathsf{K}_{Coercer}\left(\bigvee_{v \in A} \neg \mathsf{vote}_{v,1}\right)\right)\right)$$

$$\Phi_2 \equiv \langle\langle Coercer \rangle\rangle \mathbf{F} \left(\mathsf{finished} \wedge \left(\bigwedge_{v \in A} \neg \mathsf{vote}_{v,1} \rightarrow \bigvee_{v \in A} \mathsf{K}_{Coercer}(\neg \mathsf{vote}_{v,1})\right)\right)$$

$$\Phi_3 \equiv \langle\langle Coercer \rangle\rangle \mathbf{F} \left(\mathsf{finished} \wedge \left(\bigvee_{v \in A} \neg \mathsf{vote}_{v,1} \rightarrow \mathsf{K}_{Coercer}\left(\bigvee_{v \in A} \neg \mathsf{vote}_{v,1}\right)\right)\right)$$

$$\Phi_4 \equiv \langle\langle Coercer \rangle\rangle \mathbf{F} \left(\mathsf{finished} \wedge \left(\bigvee_{v \in A} \neg \mathsf{vote}_{v,1} \rightarrow \bigvee_{v \in A} \mathsf{K}_{Coercer}(\neg \mathsf{vote}_{v,1})\right)\right)$$

Figure 9.3: Formulae for Selene model checking

Let us consider a coercer attempting to force a group of voters $A \subseteq \mathbb{A}\mathrm{gt}$ to vote for his preferred candidate. We can assume w.l.o.g. that the number of the candidate is 1. The formulae in Figure 9.3 express different "flavors" of the coercer's coercive ability, with the following reading:

- Φ_1 expresses that the coercer can enforce a state where the elections are over and, if no one in A followed his orders, then the coercer knows that at least one of them disobeyed (but does not necessarily know who);
- According to Φ_2, if no one in A followed his orders, the coercer will know who for at least one of them;
- Φ_3 says that if some of the voters did not vote as ordered, then the coercer will know about it;
- Φ_4 says that if some of the voters did not vote as ordered, then the coercer will identify at least one of them.

Conceptually, the formulae capture the extent to which the coercer can identify the disobedience of the coerced voters, and hence knows when to execute his threats. Note that, when $A = \{v\}$, i.e., the coerced group consists of a single voter, then all four formulae are equivalent.

9.7.1 Results

We collect the results of the evaluation for each of the specified formulae in tables presented in Figures 9.22 to 9.25. We show performance results for the approximation

configuration	#states	tgen	Lower approx.		Upper approx.		Approx.	Exact	
			tverif	result	tverif	result	result	(tg+tv)	result
(2,1,1,1,1)	427	<1	<1	False	<1	True	?	<1	True
(2,1,1,4,4)	16777	1	<1	False	<1	True	?	249	True
(2,1,2,4,4)	22365	1	<1	True	<1	True	True	timeout	
(2,2,1,4,4)	331441	19	1	False	16	True	?	timeout	
(2,2,2,4,4)	596577	36	2	True	31	True	True	timeout	
(3,2,1,1,1)	281968	40	<1	False	4	True	?	timeout	
(4,1,1,1,1)	146001	2	<1	False	2	True	?	timeout	
(4,2,1,1,1)	memout							timeout	

Table 9.22: Experimental results for formula Φ_1 by fixpoint approximation

configuration	#states	tgen	Lower approx.		Upper approx.		Approx.	Exact	
			tverif	result	tverif	result	result	(tg+tv)	result
(2,1,1,1,1)	427	<1	<1	False	<1	True	?	<1	True
(2,1,1,4,4)	16777	1	<1	False	1	True	?	249	True
(2,1,2,4,4)	22365	1	<1	True	1	True	True	timeout	
(2,2,1,4,4)	331441	19	1	False	17	True	?	timeout	
(2,2,2,4,4)	596577	36	1	True	31	True	True	timeout	
(3,2,1,1,1)	281968	40	<1	False	5	True	?	timeout	
(4,1,1,1,1)	146001	2	<1	False	2	True	?	timeout	
(4,2,1,1,1)	memout							timeout	

Table 9.23: Experimental results for formula Φ_2 by fixpoint approximation

algorithms, both for the lower and the upper bound, and compare them to the exact verification done with MCMAS. Each row in a table corresponds to a single run of an experiment over the selected model. The columns contain the following information:

- the parameters of the model (*configuration*), consisting of the numbers of voters and available candidates, the maximal numbers of voters that the coercer can try to coerce and the clock steps that the system waits for incoming votes and for notifications from the voters about coercion attempts. E.g., configuration (2,2,2,4,4) describes the model with 2 voters, 2 candidates, the coercer coercing up to 2 voters, and the maximal time units for the system to wait for votes and

configuration	#states	tgen	Lower approx.		Upper approx.		Approx.	Exact	
			tverif	result	tverif	result	result	(tg+tv)	result
(2, 1, 1, 1, 1)	427	<1	<1	False	<1	True	?	<1	True
(2, 1, 1, 4, 4)	16777	1	<1	False	<1	True	?	250	True
(2, 1, 2, 4, 4)	22365	1	<1	True	1	True	True	timeout	
(2, 2, 1, 4, 4)	331441	19	1	False	16	True	?	timeout	
(2, 2, 2, 4, 4)	596577	36	2	True	30	True	True	timeout	
(3, 2, 1, 1, 1)	281968	40	1	False	5	True	?	timeout	
(4, 1, 1, 1, 1)	146001	2	<1	False	2	True	?	timeout	
(4, 2, 1, 1, 1)	memout							timeout	

Table 9.24: Experimental results for formula Φ_3 by fixpoint approximation

configuration	#states	tgen	Lower approx.		Upper approx.		Approx.	Exact	
			tverif	result	tverif	result	result	(tg+tv)	result
(2, 1, 1, 1, 1)	427	<1	<1	False	<1	True	?	<1	True
(2, 1, 1, 4, 4)	16777	1	<1	False	1	True	?	257	True
(2, 1, 2, 4, 4)	22365	1	<1	True	<1	True	True	timeout	
(2, 2, 1, 4, 4)	331441	19	1	False	4	False	False	timeout	
(2, 2, 2, 4, 4)	596577	36	1	False	7	False	False	timeout	
(3, 2, 1, 1, 1)	281968	40	<1	False	1	False	False	timeout	
(4, 1, 1, 1, 1)	146001	2	<1	False	3	True	?	timeout	
(4, 2, 1, 1, 1)	memout							timeout	

Table 9.25: Experimental results for formula Φ_4 by fixpoint approximation

coercion notifications being both set to 4;

- The size of the state space (*#states*) and the time that the algorithm spent on generating the data structures for the model (*tgen*);
- The running time and output of the verification algorithm (*tver, result*) for model checking the lower approximation $tr_L(\varphi)$, and similarly for the upper approximation $tr_U(\varphi)$;
- The result of the approximation (*Approx. result*), with "?" in case of inconclusive output;

configuration	#states	tgen	result	#vote	#rvote	pvote = $\frac{\#rvote}{\#vote} \times 100\%$
$(2,2,1,4,4)$	331441	14	False	200	50	25%
$(2,2,2,1,1)$	9651	<1	False	144	36	25%
$(2,2,2,4,4)$	596577	24	False	360	90	25%
$(2,3,1,2,2)$	289423	10	False	378	168	44%
$(2,3,2,1,1)$	64829	1	False	576	256	44%
$(2,3,2,2,2)$	661501	22	False	864	384	44%
$(2,3,2,2,2)$	281968	15	False	672	84	12%
$(2,3,2,2,2)$	765232	41	False	1824	228	12%
$(4,1,1,1,1)$	146001	2	False	240	0	0%

Figure 9.4: Experimental results for formula $\Phi_2^{dist_{A'}}$ with percentage coverage

- The total running time (*tg+tv*) and the result (*result*) of the exact $\mathbf{ATL}_{\text{ir}}$ model checking with MCMAS.

The running times are given in seconds. *Timeout* indicates that the process did not terminate in 2 hours. *Memout* indicates that the process is terminated by the system due to allocating too much memory.

The exact $\mathbf{ATL}_{\text{ir}}$ model checking is performed with MCMAS 1.3.0. To perform the approximate verification, we used the explicit representations of models from Section 7.6.2, and an implementation of the fixpoint algorithms from Chapter 3 in a standalone tool written in C++. The models used in both approaches were isomorphic. The tests were conducted on a Intel Core i7-6700 CPU with dynamic clock speed of 2.60 – 3.50 GHz, 32 GB RAM, running 64bit Ubuntu 16.04 Linux.

9.7.2 Counting Coercion-Friendly Configurations

The validity of a property is a strong result: if a formula is true in the model, then the coercer has a strategy to achieve his goal under all possible circumstances. The system is therefore completely insecure against the considered type of attack. If, however, the formula turns out false, it does not mean that the system is always able to defend itself. In such case we only know that there is no uniform strategy that allows the coercer to break the system's defenses, given no information about the initial state of affairs (e.g., a partially uncovered choice function). We thus attempt to quantitatively estimate the extent to which our model is safe from the attacks expressed by Φ_2. To this end

$$\Phi_2^{dist_{A'}} \equiv \langle\langle Coercer \rangle\rangle \mathbf{F} \left(\mathsf{finished} \wedge ((\bigvee_{v \in A} \neg \mathsf{vote}_{v,1} \wedge \bigwedge_{v' \in Agents \setminus A} dist_{A'}(v')) \rightarrow \bigvee_{v \in A} \mathsf{K}_{Coercer}(\neg \mathsf{vote}_{v,1})\right)$$

Figure 9.5: A formula for quantitative analysis

we inspect in detail all the possible distributions $D_{A'}$ of votes of voters outside of the coerced group A and check under which of these the coercer can precisely point to a disobedient voter. Formally, $dist_{A'} \in D_{A'}$ iff $dist_{A'}$ is a function from $Agents \setminus A$ to $Props$ such that for each $v \in Agents \setminus A$ there exists $1 \leq i \leq k$ such that $dist_{A'}(v) = \mathsf{vote}_{\mathsf{v,i}}$.

To perform quantitative analysis we utilize the formula $\Phi_2^{dist_{A'}}$ presented in Figure 9.5. Note that the formula depends on $dist_{A'} \in D_{A'}$ and $A \subseteq Agents$. Intuitively, it expresses the ability of the coercer to enforce that if some of the agents in A did not vote for candidate 1 and the remaining voters voted according to $dist_{A'}$, then the coercer can identify a voter in A that did not vote for 1.

The process of quantitative analysis is performed as follows. For a given model configuration, we fix an arbitrary coalition A. Then, for each distribution $dist_{A'} \in D_{A'}$ the formula $\Phi_2^{dist_{A'}}$ is verified. Our approach is based on state-labelling, hence we can inspect all the states reached just after publishing votes. In Fig. 9.4 by $\#vote$ we denote the aggregate number of such states that are consistent with any $dist_{A'} \in D_{A'}$ and $\#rvote$ collects the count of how many of these states satisfy $\Phi_2^{dist_{A'}}$. As we can observe, there are cases where the coercer can gain advantage in nearly half of considered distributions.

It should be emphasized that approximate algorithms are used for model checking $\Phi_2^{dist_{A'}}$. Thus, the *pvote* shows only the percentage of *confirmed cases* where a successful coercion strategy exists. The actual counts may be larger, since the approximations provide only guaranteed lower bound estimation.

9.7.3 Discussion of Results

As confirmed by the experiments, the question posed by formula Φ_2 is the most restrictive. Namely, in Φ_2 we ask whether the coercer has a general strategy to find out exactly which voter voted against his demands, assuming that there was a disobedient one. The answer to this question is true only in special cases of a single candidate. On

the other hand, the results of verification of Φ_1 reveal that the system is sometimes not able to fully defend a coerced group and the coercer can detect that at least one of the members did not follow the demands. To illustrate this on a simple example, in a model of two voters and two ballots the coercer has a trivial strategy: request a vote for candidate 1 from both the voters. Moreover, in this case the coercer has even more knowledge, as he knows which one of the voters deceived (they both did).

Exact model checking with MCMAS seems infeasible in most of the cases, except for the small models up to hundreds of states. The approximations, in turn, offer a dramatic speedup, enabling verification of models up to hundreds of thousands of states. Although the approximate method is faster, the results can be inconclusive; namely, we observe cases where the truth value of $tr_L(\varphi)$ differs from the value of $tr_U(\varphi)$. It should be noted that in the approximate approach the graphs are represented explicitely in memory, unlike in the case of BDD-based symbolic methods. Still, memory is cheaper and easier to buy than time.

9.8 Social Explainable AI (SAI)

The scalable class of models is discussed in Section 7.4.2. During our model checking experiments, we employed two system specification variants: one allowing for an impersonation attack, and the other permitting a man-in-the-middle attack. We conducted verifications for the following formulas:

- $\varphi_1 \equiv \langle\langle I \rangle\rangle \mathbf{G}(shared_p \rightarrow (\bigwedge_{i \in [1,n]} mqual_i \leq k))$
- $\varphi_2 \equiv \langle\langle I \rangle\rangle \mathbf{G}(shared_p \rightarrow (\bigvee_{i \in [1,n]} mqual_i \leq k))$

Formula φ_1 assesses whether the Intruder has a strategy to ensure that all honest agents fail to achieve a quality higher than k. Conversely, φ_2 examines the feasibility of this outcome for at least one agent.

9.8.1 Results

We detail the verification results in Figures 9.6 and 9.7. The column #**Ag** denotes the scalability factor, representing the number of agents within the system. Columns #**st** and #**tr** enumerate the total number of global states and transitions, respectively,

#Ag	#st	#tr	Gen	Verif φ_1	Verif φ_2
2	886	2007	< 0.1	< 0.1/False	< 0.1/True
3	79806	273548	28	151/False	202/True
4	6538103	29471247	1284	5061/False	5102/True
5	93581930	623680431	7845	25828/False	25916/True
6	timeout				

Figure 9.6: Verification results for the Impersonator attack by fixpoint approximation

#Ag	#st	#tr	Gen	Verif φ_1	Verif φ_2
3	23966	67666	12	21/False	33/True
4	4798302	20257664	875	3810/False	3882/True
5	71529973	503249452	5688	19074/False	20103/True
6	timeout				

Figure 9.7: Verification results for Man in the Middle attack by fixpoint approximation

in the derived system model. The column **Gen** indicates the time required for model generation. The verification times and outcomes for formulas φ_1 and φ_2 are displayed under **Verif** φ_1 and **Verif** φ_2, respectively. All time measurements are expressed in seconds. A timeout threshold was established at 8 hours.

9.8.2 Discussion of Results

We successfully verified models of SAI for systems comprising up to 5 agents. The verification outcomes were conclusive across all instances; the model checker invariably returned either `True` or `False`. This achievement implies that we effectively conducted model checking for systems with nearly a billion transitions, a notable accomplishment given the **NP**-hard nature of the verification problem. In every evaluated case, the formula φ_2 was consistently validated as true. This outcome indicates that both impersonation and man-in-the-middle attacks can indeed interfere with the learning process, thereby hindering certain agents from acquiring high-quality PAIVs. Conversely, φ_1 was universally determined to be false. Therefore, it is clear that an intruder cannot compromise *all* PAIVs, even when deploying the most effective attack strategies.

9.9 Challenges and Lessons Learnt

Modeling Complex Systems The formal verification of real-world systems revealed three fundamental challenges:

1. **Abstraction-Completeness Trade-off**: Effective modeling requires balancing abstraction with semantic fidelity. The bridge card game analysis demonstrated how over-abstraction (e.g., omitting trick resolution dynamics) could compromise verification of strategic reasoning under partial observability, while excessive detail in drone coordination models led to state-space explosion in systems with $N > 10$ agents.

2. **Temporal-Epistemic Complexity**: The SAI framework highlighted the exponential growth of epistemic equivalence classes. This necessitated optimized group-theoretic verification algorithms to maintain computational tractability.

3. **Security-Centric Modeling**: Coercion-resistant voting protocols like Selene required novel formalizations of strategic-epistemic properties. The implementation of $K_v(\text{voted}_i \vee \neg\text{voted}_i)$ specifications demanded sophisticated state-space partitioning to prevent trivial satisfaction through passive proposition signaling.

Verification Methodology Insights Our multi-agent system analyses yielded key methodological advancements:

1. **Structured Abstraction Frameworks**: The bridge endplay model established best practices for state-space reduction through:

   - Hierarchical state representation with $(\text{hands}, \text{tricks}, \text{board}, \text{clock})$ tuples
   - Contextual goal structures separating verification objectives from environmental dynamics

2. **Parametric Scalability Mechanisms**: Drone coordination models demonstrated the effectiveness of sublinear complexity functions $f_{\text{epist}}(N)$ and $f_{\text{win}}(N)$ in managing state-space explosion. This enabled verification of systems through controlled topological parameterization.

3. **Security-by-Design Principles**: The Selene analysis emphasized the importance of integrating security properties at the modeling stage. Key innovations included:

   - Cryptographic commitment tracking through SN $\rightarrow$ WBB mappings
   - Robust anti-coercion spaces $E(B)$ with combinatorial indistinguishability guarantees

10

Conclusions

Modal logics for Multi-Agent Systems (MAS), along with their associated verification problems, have been a subject of extensive study for decades. These logics are inherently marked by high computational complexity, a trait that manifests both in theoretical analysis and practical application. Despite these challenges, it is essential to engage in the verification of complex, real-life scenarios to drive meaningful advancements in the field.

In this thesis, we have delved into the intricate problem of verifying strategic abilities within the context of imperfect information. We have made significant strides by proposing, developing, and evaluating a suite of model-checking algorithms, often optimizing them for enhanced performance and efficiency. In parallel, we have conducted thorough studies on various e-voting protocols and complex real-life scenarios. Our efforts in modeling these scenarios, defining critical properties, and applying our developed algorithms for verification have been instrumental in pushing the boundaries of what is achievable in this domain.

The inherent challenges of this endeavor, notably the high computational demands and the prevalent issue of state-space explosion, are formidable. Yet, our research demonstrates that these obstacles can be surmounted in several instances. We have shown that with innovative algorithmic strategies and a deep understanding of the specificities of each scenario, effective verification of strategic properties in MAS under conditions of imperfect information is not only possible but also achievable in practice.

Moreover, our work contributes to bridging the gap between theoretical constructs and their practical applications. By applying our findings to real-world scenarios, par-

ticularly in the realm of e-voting, we have underscored the relevance and applicability of modal logics in MAS. This not only validates the theoretical framework but also showcases its utility in addressing complex, real-world problems.

In conclusion, while the journey towards fully efficient and comprehensive verification in MAS is ongoing, the contributions of this thesis represent significant steps forward. The methodologies and insights presented in this work lay a foundation for future research, opening avenues for further exploration and refinement in verification of Multi-Agent Systems.

The course of this research has elucidated several critical insights into the complexities of model checking under conditions of imperfect information and recall:

1. **Complexity Origins**: The challenge of model checking in scenarios of imperfect information and recall is not solely attributed to the presence of epistemic classes but is also significantly compounded by the state-space explosion phenomenon. This complexity necessitates innovative approaches to verification.

2. **Epistemic Relation Handling**: Vigilance in managing epistemic relations is imperative during the verification phase to prevent overlooking subtle agent interactions, which could lead to the development of conflicting strategies.

3. **Algorithm Design Focus**: When designing model checking algorithms, there is an inherent trade-off between optimizing time complexity and space complexity. A strategic approach is essential, as optimizing for both concurrently is typically unfeasible.

4. **Abstractions and Heuristics**: The application of abstractions and heuristics can significantly streamline the verification process. However, their selection must be meticulously executed to maintain the accuracy and reliability of verification outcomes.

5. **Model Design Importance**: The precision in model design is paramount, rivaling the significance of the verification algorithm itself. Tools that feature graphical interfaces for model representation, such as transition graphs, can substantially enhance model accuracy and verification validity.

6. **Challenges with ATL Formulas**: Despite the initial intuitive appeal of ATL formulas, their implementation presents notable challenges. It is deceptively easy to devise a formula that appears valid but fails to accurately capture the intended dynamics or behaviors.

As the main takeaway message I propose the following summary. The essence of tackling a verification problem lies in a methodical approach. Initially, it is crucial to identify the pertinent agents, their actions, their knowledge, and, most critically, their interactions with each other and the environment. This identification process lays the groundwork for designing a model that encapsulates only the relevant aspects of the scenario at hand. Following this, the next step involves the meticulous formulation of the intended properties, which are then captured using `ATL` formulas. Subsequently, selecting an appropriate model checking algorithm and applying it to the designed model is imperative. Should the outcomes diverge from expectations, it signals a potential flaw in either the model or the formula. In instances where the model's complexity hinders verification, employing partial order reduction methods can offer a solution. If these methods prove insufficient, resorting to model abstraction may provide the necessary simplification for effective verification.

References

T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge. Knowledge and ability. In H.P. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B.P. Kooi, editors, *Handbook of Epistemic Logic*, pages 543–589. College Publications, 2015. 153

Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997. 17

Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time Temporal Logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998. 29

Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002. doi: 10.1145/585265.585270. 14, 19, 25, 26, 44, 54, 55

Francesco B. and A. Lomuscio. Agent-based abstractions for verifying alternating-time temporal logic with imperfect information. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems AAMAS*, pages 1259–1267. IFAAMAS, 2017. 96

T. Ball and O. Kupferman. An abstraction-refinement framework for multi-agent systems. In *Proceedings of Logic in Computer Science (LICS)*, pages 379–388, 2006. doi: 10.1109/LICS.2006.10. 96

G. Behrmann, A. David, and K.G. Larsen. A tutorial on UPPAAL. In *Formal Methods for the Design of Real-Time Systems: SFM-RT*, number 3185 in LNCS, pages 200–236. Springer, 2004. 198

F. Belardinelli, R. Condurache, C. Dima, W. Jamroga, and A.V. Jones. Bisimulations for verification of strategic abilities with application to ThreeBallot voting protocol. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1286–1295. IFAAMAS, 2017a. 80

Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Verification of multi-agent systems with imperfect information and public actions. In *Proceedings of AAMAS*, pages 1268–1276, 2017b. 55

Francesco Belardinelli, Wojciech Jamroga, Damian Kurpiewski, Vadim Malvone, and Aniello Murano. Strategy logic with simple goals: Tractable reasoning about strategies. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI*, pages 88–94, 2019. doi: 10.24963/ijcai.2019/13. 11

N. Belnap and M. Perloff. Seeing to it that: a canonical form for agentives. *Theoria*, 54:175–199, 1988. 14

D. Berwanger and L. Kaiser. Information tracking in games on graphs. *Journal of Logic, Language and Information*, 19(4):395–412, 2010. doi: 10.1007/s10849-009-9115-8. 55

D. Berwanger, A.B. Mathew, and M. van den Bogaard. Hierarchical information patterns and distributed strategy synthesis. In *Proceedings of ATVA*, volume 9364 of *Lecture Notes in Computer Science*, pages 378–393. Springer, 2015. doi: 10.1007/978-3-319-24953-7_28. 55

A. Bruni, E. Drewsen, and C. Schürmann. Towards a mechanized proof of Selene receipt-freeness and vote-privacy. In *Proceedings of E-Vote-ID*, volume 10615 of *Lecture Notes in Computer Science*, pages 110–126. Springer, 2017. doi: 10.1007/978-3-319-68687-5_7. 35

G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Proceedings of Computer Aided Verification (CAV)*, pages 274–287, 1999. doi: 10.1007/3-540-48683-6_25. 96

N. Bulling and W. Jamroga. Alternating epistemic mucalculus. In *Proceedings of IJCAI-11*, pages 109–114, 2011. 26, 28

N. Bulling and W. Jamroga. Comparing variants of strategic ability: How uncertainty and memory influence general properties of games. *Journal of Autonomous Agents and Multi-Agent Systems*, 28(3):474–518, 2014. 20

S. Busard. *Symbolic Model Checking of Multi-Modal Logics: Uniform Strategies and Rich Explanations*. PhD thesis, Universite Catholique de Louvain, 2017. 91

Simon Busard, Charles Pecheur, Hongyang Qu, and Franco Raimondi. Improving the model checking of strategies under partial observability and fairness constraints. In *Formal Methods and Software Engineering*, volume 8829 of *Lecture Notes in Computer Science*, pages 27–42. Springer, 2014. ISBN 978-3-319-11736-2. doi: 10.1007/978-3-319-11737-9_3. 156

Simon Busard, Charles Pecheur, Hongyang Qu, and Franco Raimondi. Reasoning about memoryless strategies under partial observability and unconditional fairness constraints. *Information and Computation*, 242:128–156, 2015. doi: 10.1016/j.ic.2015.03.014. 91

K. Chatterjee, L. Doyen, T.A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games of incomplete information. *Logical Methods in Computer Science*, 3(3), 2007. 54

A. Cimatti, E.M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, M. Sebastiani, and A Tacchella. NuSMV2: An open-source tool for symbolic model checking. In *Proceedings of Computer Aided Verification (CAV)*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364, 2002. 35

E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994. 96

REFERENCES

Marco Conti and Andrea Passarella. The internet of people: A human and data-centric paradigm for the next generation internet. *Comput. Commun.*, 131:51–65, 2018. doi: 10.1016/j.comcom.2018.07.034. 132

Pierluigi Contucci, Janos Kertesz, and Godwin Osabutey. Human-ai ecosystem with abrupt changes as a function of the composition. *PLOS ONE*, 17(5):1–12, 05 2022. doi: 10.1371/journal.pone.0267310. 132, 133

P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages*, pages 238–252, 1977. doi: 10.1145/512950.512973. 96

P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Półrola, M. Szreter, B. Woźna, and A. Zbrzezny. Verics: A tool for verifying timed automata and estelle specifications. In *Proceedings of the of the 9th Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *Lecture Notes in Computer Science*, pages 278–283. Springer, 2003. 33

C. Dima, B. Maubert, and S. Pinchinat. The expressive power of epistemic μ-calculus. *CoRR*, abs/1407.5166, 2014. 28

C. Dima, B. Maubert, and S. Pinchinat. Relating paths in transition systems: The fall of the modal mu-calculus. In *Proceedings of Mathematical Foundations of Computer Science (MFCS)*, volume 9234 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2015. doi: 10.1007/978-3-662-48057-1_14. 28

Georgios Drainakis, Konstantinos V. Katsaros, Panagiotis Pantazopoulos, Vasilis Sourlas, and Angelos Amditis. Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis. In *Proceedings of NCA*, pages 1–8. IEEE, 2020. doi: 10.1109/NCA51143.2020.9306745. 132

Andrew Fuchs, Andrea Passarella, and Marco Conti. Modeling human behavior part I - learning and belief approaches. *CoRR*, abs/2205.06485, 2022. doi: 10.48550/arXiv.2205.06485. 132, 133

B.A. Galler and M.J. Fisher. An improved equivalence algorithm. *Communications of the ACM*, 7(5):301–303, 1964. doi: 10.1145/364099.364331. 57, 59

P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proc. of the 16th Int. Conf. on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004. 34

P. Godefroid and R. Jagadeesan. Automatic abstraction using generalized model checking. In *Proceedings of Computer Aided Verification (CAV)*, volume 2404 of *Lecture Notes in Computer Science*, pages 137–150. Springer, 2002. doi: 10.1007/3-540-45657-0_11. 96

Dimitar P. Guelev, Catalin Dima, and Constantin Enea. An alternating-time temporal logic with knowledge, perfect recall and past: axiomatisation and model-checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011. 55

István Hegedüs, Gábor Danner, and Márk Jelasity. Gossip learning as a decentralized alternative to federated learning. In *Proceedings of IFIP DAIS*, volume 11534 of *Lecture Notes in Computer Science*, pages 74–90. Springer, 2019. doi: 10.1007/978-3-030-22496-7_5. 133

István Hegedüs, Gábor Danner, and Márk Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *J. Parallel Distributed Comput.*, 148:109–124, 2021. doi: 10.1016/j.jpdc.2020.10.006. 133

W. Jamroga and T. Ågotnes. Constructive knowledge: What agents can achieve under incomplete information. *Journal of Applied Non-Classical Logics*, 17(4):423–475, 2007. 29

W. Jamroga and J. Dix. Model checking ATL$_{ir}$ is indeed Δ_2^P-complete. In *Proceedings of EUMAS*, volume 223 of *CEUR Workshop Proceedings*, 2006. 19

W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3):185–219, 2004. 20, 153

W. Jamroga, M. Knapik, and D. Kurpiewski. Fixpoint approximation of strategic abilities under imperfect information. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1241–1249. IFAAMAS, 2017a. 11, 80

W. Jamroga, V. Malvone, and A. Murano. Reasoning about natural strategic ability. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 714–722. IFAAMAS, 2017b. 28

W. Jamroga, M. Knapik, and D. Kurpiewski. Model checking the SELENE e-voting protocol in multi-agent logics. In *Proceedings of the 3rd International Joint Conference on Electronic Voting (E-VOTE-ID)*, volume 11143 of *Lecture Notes in Computer Science*, pages 100–116. Springer, 2018a. 11

W. Jamroga, W. Penczek, P. Dembiński, and A. Mazurkiewicz. Towards partial order reductions for strategic ability. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 156–165. IFAAMAS, 2018b. 103, 104, 105

W. Jamroga, W. Penczek, and T. Sidoruk. Strategic abilities of asynchronous agents: Semantic paradoxes and how to tame them. *CoRR*, abs/2003.03867, 2020a. URL `https://arxiv.org/abs/2003.03867`. 103, 105

Wojciech Jamroga and Damian Kurpiewski. Pretty good strategies and where to find them. In Vadim Malvone and Aniello Murano, editors, *Multi-Agent Systems - 20th European Conference, EUMAS 2023, Naples, Italy, September 14-15, 2023, Proceedings*, volume 14282 of *Lecture Notes in Computer Science*, pages 363–380. Springer, 2023. doi: 10.1007/978-3-031-43264-4_23. URL `https://doi.org/10.1007/978-3-031-43264-4_23`. 5, 86, 112, 176

Wojciech Jamroga, Michał Knapik, Damian Kurpiewski, and Łukasz Mikulski. Approximate verification of strategic abilities under imperfect information. *Artificial Intelligence*, 277, 2019a. doi: 10.1016/j.artint.2019.103172. 7, 13, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 91, 96, 100, 101, 102, 112, 176

Wojciech Jamroga, Vadim Malvone, and Aniello Murano. Natural strategic ability. *Artificial Intelligence*, 277, 2019b. doi: 10.1016/j.artint.2019.103170. 28

Wojciech Jamroga, Yan Kim, Damian Kurpiewski, and Peter Y. A. Ryan. Towards model checking of voting protocols in uppaal. In *Proceedings of E-Vote-ID*, volume 12455 of *Lecture Notes in Computer Science*, pages 129–146. Springer, 2020b. doi: 10.1007/978-3-030-60347-2_9. 7, 13

Wojciech Jamroga, Beata Konikowska, Wojciech Penczek, and Damian Kurpiewski. Multi-valued verification of strategic ability. *Fundamenta Informaticae*, 175(1-4):207–251, 2020c. doi: 10.3233/FI-2020-1955. 11

Wojciech Jamroga, Damian Kurpiewski, and Vadim Malvone. Natural strategic abilities in voting protocols. In Thomas Groß and Luca Viganò, editors, *Socio-Technical Aspects in Security and Trust - 10th International Workshop, STAST 2020, Virtual Event, September 14, 2020, Revised Selected Papers*, volume 12812 of *Lecture Notes in Computer Science*, pages 45–62. Springer, 2020d. doi: 10.1007/978-3-030-79318-0_3. URL https://doi.org/10.1007/978-3-030-79318-0_3. 11

Wojciech Jamroga, Wojciech Penczek, and Teofil Sidoruk. Strategic abilities of asynchronous agents: Semantic side effects and how to tame them. In *Proceedings of KR 2021*, pages 368–378, 2021. 22, 24, 25

Wojciech Jamroga, Damian Kurpiewski, and Vadim Malvone. How to measure usable security: Natural strategies in voting protocols. *J. Comput. Secur.*, 30(3):381–409, 2022a. doi: 10.3233/JCS-210049. URL https://doi.org/10.3233/JCS-210049. 6, 13, 112, 176

Wojciech Jamroga, Yan Kim, and Damian Kurpiewski. Scalable verification of social explainable AI by variable abstraction. In Ana Paula Rocha, Luc Steels, and H. Jaap van den Herik, editors, *Proceedings of the 16th International Conference on Agents and Artificial Intelligence, ICAART 2024, Volume 1, Rome, Italy, February 24-26, 2024*, pages 149–158. SCITEPRESS, 2024. doi: 10.5220/0012474800003636. URL https://doi.org/10.5220/0012474800003636. 10

Wojtek Jamroga, Lukasz Masko, Lukasz Mikulski, Witold Pazderski, Wojciech Penczek, Teofil Sidoruk, and Damian Kurpiewski. Verification of multi-agent properties in electronic voting: A case study. In David Fernández-Duque, Alessandra Palmigiano, and Sophie Pinchinat, editors, *Advances in Modal Logic, AiML 2022, Rennes, France, August 22-25, 2022*, pages 531–556. College Publications, 2022b. 5, 73, 96, 105, 112, 176

Mateusz Kaminski, Damian Kurpiewski, and Wojciech Jamroga. STV+KH: towards practical verification of strategic ability for knowledge and information flow. In Mehdi Dastani, Jaime Simão Sichman, Natasha Alechina, and Virginia Dignum, editors, *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024*, pages 2812–2814. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024. doi: 10.5555/3635637.3663296. URL https://dl.acm.org/doi/10.5555/3635637.3663296. 9

Mateusz Kaminski, Damian Kurpiewski, and Wojciech Jamroga. Natstv: Towards verification of natural strategic ability. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 11072–11076. ijcai.org, 2025. doi: 10.24963/IJCAI.2025/1265. URL https://doi.org/10.24963/ijcai.2025/1265. 9

Damian Kurpiewski and Diego Marmsoler. Strategic logics for collaborative embedded systems. *SICS Softw.-Intensive Cyber Phys. Syst.*, 34(4):201–212, 2019. doi: 10.1007/S00450-019-00424-7. URL https://doi.org/10.1007/s00450-019-00424-7. 8, 112, 176

Damian Kurpiewski, Wojciech Jamroga, and Michał Knapik. STV: Model checking for strategies under imperfect information. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*, pages 2372–2374. IFAAMAS, 2019a. 8, 159

Damian Kurpiewski, Michał Knapik, and Wojciech Jamroga. On domination and control in strategic ability. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*, pages 197–205. IFAAMAS, 2019b. 7, 73, 75, 78, 79, 80, 83, 86, 91, 112, 176

Damian Kurpiewski, Michal Knapik, and Wojciech Jamroga. On domination and control in strategic ability (extended abstract). In *Proceedings of the 31st Benelux Conference on Artificial Intelligence (BNAIC)*, 2019c. 11

Damian Kurpiewski, Witold Pazderski, Wojciech Jamroga, and Yan Kim. STV+Reductions: Towards practical verification of strategic ability using model reductions. In *Proceedings of AAMAS*, pages 1770–1772. ACM, 2021. 6, 159

Damian Kurpiewski, Lukasz Mikulski, and Wojciech Jamroga. STV+AGR: towards verification of strategic ability using assume-guarantee reasoning. In Reyhan Aydogan, Natalia Criado, Jérôme Lang, Víctor Sánchez-Anguix, and Marc Serramia, editors, *PRIMA 2022: Principles and Practice of Multi-Agent Systems - 24th International Conference, Valencia, Spain, November 16-18, 2022, Proceedings*, volume 13753 of *Lecture Notes in Computer Science*, pages 691–696. Springer, 2022. doi: 10.1007/978-3-031-21203-1_47. URL https://doi.org/10.1007/978-3-031-21203-1_47. 10

Damian Kurpiewski, Wojciech Jamroga, and Teofil Sidoruk. Towards modelling and verification of social explainable AI. In Ana Paula Rocha, Luc Steels, and H. Jaap van den Herik, editors, *Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 1, Lisbon, Portugal, February 22-24, 2023*, pages 396–403. SCITEPRESS, 2023. doi: 10.5220/0011799900003393. URL https://doi.org/10.5220/0011799900003393. 5, 112, 176

Damian Kurpiewski, Mateusz Kaminski, and Wojciech Jamroga. STV+FLY: on-the-fly model checking of strategic ability in multi-agent systems. In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz, editors, *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024,*

REFERENCES

Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024), volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 4483–4486. IOS Press, 2024. doi: 10.3233/FAIA241035. URL https://doi.org/10.3233/FAIA241035. 9

Damian Kurpiewski, Wojciech Jamroga, and Yan Kim. Approximate verification of strategic abilities under imperfect information using local models. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 143–151. ijcai.org, 2025. doi: 10.24963/IJCAI.2025/17. URL https://doi.org/10.24963/ijcai.2025/17. 4

M. Kwiatkowska, G. Norman, and D. Parker. PRISM: probabilistic symbolic model checker. In *Proceedings of TOOLS*, volume 2324 of *Lecture Notes in Computer Science*, pages 200–204. Springer, 2002. doi: 10.1007/3-540-46029-2_13. 33

A. Lomuscio and F. Raimondi. Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 161–168, 2006. doi: 10.1145/1160633.1160660. 91

Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 19(1):9–30, 2017. doi: 10.1007/s10009-015-0378-x. 31, 91

Valerio Lorenzo, Chiara Boldrini, and Andrea Passarella. SAI simulator for social AI gossiping, 2022. https://zenodo.org/record/5780042. 133

J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. 14

S. Meier, B. Schmidt, C. Cremers, and D.A. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *Computer Aided Verification, Proceedings of CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013. doi: 10.1007/978-3-642-39799-8_48. 35

Łukasz Mikulski, Wojciech Jamroga, and Damian Kurpiewski. Towards assume-guarantee verification of strategic ability. In *Proc. of AAMAS'22*, pages 1702–1704. IFAAMAS, 2022a. 10

Lukasz Mikulski, Wojciech Jamroga, and Damian Kurpiewski. Assume-guarantee verification of strategic ability. In Reyhan Aydogan, Natalia Criado, Jérôme Lang, Víctor Sánchez-Anguix, and Marc Serramia, editors, *PRIMA 2022: Principles and Practice of Multi-Agent Systems - 24th International Conference, Valencia, Spain, November 16-18, 2022, Proceedings*, volume 13753 of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2022b. doi: 10.1007/978-3-031-21203-1_11. URL https://doi.org/10.1007/978-3-031-21203-1_11. 10

Lukasz Mikulski, Wojciech Jamroga, and Damian Kurpiewski. Towards assume-guarantee verification of strategic ability. In Piotr Faliszewski, Viviana Mascardi, Catherine Pelachaud, and Matthew E. Taylor, editors, *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, pages 1702–1704. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022c. doi: 10.5555/3535850.3536082. URL https://www.ifaamas.org/Proceedings/aamas2022/pdfs/p1702.pdf. 10

R.C. Moore. Reasoning about knowledge and action. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 223–227. William Kaufmann, 1977. 14

R.C. Moore. A formal theory of knowledge and action. In J. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*. Ablex Publishing Corp., 1985. 14

Abdul-Rasheed Ottun, Pramod C. Mane, Zhigang Yin, Souvik Paul, Mohan Liyanage, Jason Pridmore, Aaron Yi Ding, Rajesh Sharma, Petteri Nurmi, and Huber Flores. Social-aware federated learning: Challenges and opportunities in collaborative data training. *IEEE Internet Computing*, pages 1–7, 2022. doi: 10.1109/MIC.2022.3219263. 132

J. Pilecki, M.A. Bednarczyk, and W. Jamroga. Synthesis and verification of uniform strategies for multi-agent systems. In *Proceedings of CLIMA XV*, volume 8624 of *Lecture Notes in Computer Science*, pages 166–182. Springer, 2014. 155

J. Pilecki, M.A. Bednarczyk, and W. Jamroga. SMC: Synthesis of uniform strategies and verification of strategic ability for multi-agent systems. *Journal of Logic and Computation*, 27(7):1871–1895, 2017. doi: 10.1093/logcom/exw032. 21, 34, 91

P.Y.A. Ryan, P.B. Rønne, and V. Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In *Financial Cryptography and Data Security: Proceedings of FC 2016. Revised Selected Papers*, volume 9604 of *Lecture Notes in Computer Science*, pages 176–192. Springer, 2016. doi: 10.1007/978-3-662-53357-4_12. 143

G. Ryle. *The Concept of Mind*. Chicago University Press, 1949. 14

Bernd-Holger Schlingloff. Specification and verification of collaborative transport robots. In *4th International Workshop on Emerging Ideas and Trends in the Engineering of Cyber-Physical Systems, EITEC@CPSWeek 2018, 10 April 2018, Porto, Portugal*, pages 3–8. IEEE Computer Society, 2018. doi: 10.1109/EITEC.2018.00006. URL https://doi.org/10.1109/EITEC.2018.00006. 124

Pierre-Yves Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004. 14, 16, 19, 20, 25, 55

Social AI gossiping. Micro-project in Humane-AI-Net. Project website, 2022. https://www.ai4europe.eu/research/research-bundles/social-ai-gossiping. 133

Social Explainable AI, CHIST-ERA. Project website, 2021–24. http://www.sai-project.eu/. 132, 133

Mustafa Toprak, Chiara Boldrini, Andrea Passarella, and Marco Conti. Harnessing the power of ego network layers for link prediction in online social networks. *CoRR*, abs/2109.09190, 2021. 132

W. van der Hoek and M. Wooldridge. Cooperation, knowledge and time: Alternating-time Temporal Epistemic Logic and its applications. *Studia Logica*, 75(1):125–157, 2003. 29

Wiebe van der Hoek and Michael Wooldridge. Tractable multiagent planning for epistemic goals. In C. Castelfranchi and W.L. Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 1167–1174. ACM Press, New York, 2002. 14