NatSTV: Towards Verification of Natural Strategic Ability

Mateusz Kamiński[2,1], **Damian Kurpiewski**[1,2] and **Wojciech Jamroga**[1,2]
[1]Nicolaus Copernicus University in Toruń
[2]Institute of Computer Science, Polish Academy of Sciences
{m.kaminski, d.kurpiewski, w.jamroga}@ipipan.waw.pl

Abstract

We present **NatSTV**, a tool for approximate verification of *natural strategic ability* in multi-agent systems. The tool builds on our model checker **STV** (STrategic Verifier), and implements heuristic synthesis of natural strategies for asynchronous agents with imperfect information and recall. All of that is available through a web interface, with no need to install or configure the software by the user.

1 Introduction

Multi-agent systems (MAS) are based on the interaction of multiple autonomous agents, some of them intelligent and/or proactive, often exhibiting purposeful collaborative (or, conversely, adversarial) behavior. Formal methods for analysis and verification are essential to ascertain their functionality and security. *Alternating-time temporal logic* **ATL** [Alur *et al.*, 2002; Schobbens, 2004] is a widely recognized framework for reasoning about the *strategic abilities* of agents, that enables to address the potential outcomes of individual agents and their coalitions. Over the past two decades, substantial effort has been devoted to algorithms for formal verification of strategic logics [Alur *et al.*, 2002; Mogavero *et al.*, 2014; Chen *et al.*, 2013; Busard *et al.*, 2014; Huang and van der Meyden, 2014; Cermak *et al.*, 2014; Cermák *et al.*, 2015; Jamroga *et al.*, 2019a]. Based on those, a number of model checking tools have been developed, including Mocha [Alur *et al.*, 1998], MCMAS [Lomuscio *et al.*, 2017] and VITAMIN [Ferrando and Malvone, 2024]. However, the computational complexity of model checking for strategic abilities, especially under imperfect information, is very challenging [Bulling *et al.*, 2010; Jamroga, 2015].

To address the challenges, our group at the Polish Academy of Sciences has developed a suite of techniques for *incomplete* and *approximate verification* of **ATL** for agents with imperfect information, implemented in our experimental open-source model checker **STV** (STrategic Verifier) [Kurpiewski *et al.*, 2019; Kurpiewski *et al.*, 2021; Kaminski *et al.*, 2024; Kurpiewski *et al.*, 2024]. Here, we expand the suite with sound but incomplete verification of *natural strategic ability*, expressed in "Natural **ATL**," or **NatATL** [Jamroga *et al.*, 2019b; Jamroga *et al.*, 2019c].

The new tool, called **NatSTV**, provides a user-friendly environment for analyzing such requirements, featuring a graphical user interface (GUI) and a flexible model specification language. Thanks to that, it has significant pedagogical value, serving as an intuitive introduction to the complex subjects of strategic reasoning and model checking of strategic logics. Previous versions of **STV** have been utilized in tutorials and graduate courses at top AI conferences and summer schools, including IJCAI 2022, PRIMA 2022, ESSAI 2023, and ECAI 2024. The only other tool to synthesize and verify natural strategies is a very recent extension of VITAMIN [Aruta *et al.*, 2024]. However, it only tackles the case of perfect information strategies, whereas **NatSTV** verifies MAS with imperfect information (i.e., partial observability).

2 Application Domain

NatSTV addresses the formal verification of strategic abilities in multi-agent systems. **ATL** formulas like $\langle\langle taxi, passg\rangle\rangle\Diamond$destination ("the autonomous cab and the passenger have a joint strategy to eventually arrive at the destination") and $\langle\langle passg\rangle\rangle\Box$alive ("the passenger can keep staying alive") can be used to express important functionality, safety, and security requirements in MAS [Jamroga, 2015]. This is, e.g., relevant for specification and verification of e-voting. Properties such as anonymity, coercion-resistance, and voter verifiability are crucial for voting procedures, and have a strong strategic component. However, case studies [Jamroga *et al.*, 2018; Jamroga *et al.*, 2020b; Kurpiewski *et al.*, 2022] have demonstrated that practical verification of those properties is infeasible due to the state-, transition- and strategy-space explosion.

Natural strategies were proposed in [Jamroga *et al.*, 2019b; Jamroga *et al.*, 2019c] to capture the abilities of agents with limited computational resources, such as humans, drones, sensor networks etc. In particular, natural strategic abilities provide a more suitable semantics for reasoning about what human voters can plausibly achieve [Jamroga *et al.*, 2021a]. Extensions of **NatATL** have been used to analyze the abilities of participants in keyword auctions and probabilistic access control [Belardinelli *et al.*, 2022; Berthon *et al.*, 2024].

NatSTV is our first step towards practical verification of such abilities. Given a modular specification of a MAS and a formula $\langle\langle A\rangle\rangle\gamma$, it attempts to synthesize and simplify a natural strategy for A to enforce the temporal property γ. Then, it

reports the size k of the smallest natural strategy that has been found. Thus, for all complexity constraints of $k' \geq k$, we get that the $\mathbf{NatATL}$ formula $\langle\langle A \rangle\rangle^{\leq k'} \gamma$ holds. In this sense, **NatSTV** provides sound but not complete model checking for $\mathbf{NatATL}$ with imperfect information.

3 Formal Background

Model checking takes a specification of the system model M and a logical formula φ, and asks if φ is satisfied by M.

Modules and models. The primary input consists of a collection of asynchronous modules [Lomuscio *et al.*, 2013; Jamroga *et al.*, 2020a], where each local state consists of a *location label*, possibly with a valuation of *state variables*. Local transitions in an agent's module are determined by the current location, the choice of the agent, and possibly the valuation of input variables. They can be either private or shared between several agents. In the latter case, the transition can be executed only if all the involved agents synchronously participate in it. An example specification of an agent module in the input language of **NatSTV** is shown in Figure 1.

The *global model* of the MAS is given by the asynchronous product of its modules. In particular, global states are tuples of local states, one per agent, reachable from the initial configuration, cf. [Jamroga *et al.*, 2021b; Kurpiewski *et al.*, 2022] for the details.

General strategies. A strategy is a conditional plan that prescribes the agents' actions for every possible situation [Alur *et al.*, 2002; Schobbens, 2004]. We focus on *imperfect information memoryless strategies*, which are functions mapping the agent's local states to its available choices. The *outcome* of a strategy from state q consists of all infinite paths starting from q that are consistent with the strategy.

Natural strategies. Let $\mathcal{B}(Prop)$ denote the set of Boolean formulas over some atomic propositions $Prop$. Natural memoryless strategies for agent a are represented by ordered lists of guarded actions. That is, s_a consists of pairs $\phi_i \rightsquigarrow \alpha_i$ where $\phi_i \in \mathcal{B}(Prop)$ and α_i is a choice available to a in states where ϕ_i holds. To enforce that only uniform (i.e., executable) strategies are used, we impose that only the local atomic propositions of module a can be used within s_a. A collective strategy s_A is simply a tuple of individual strategies, one per $a \in A$.

Complexity of strategies. The complexity of a natural strategy s_A is defined as $compl(s_A) = \sum_{(\phi,\alpha)\in s_A} |\phi|$, where $|\phi|$ represents the number of symbols in the guard ϕ, excluding parentheses. Thus, $compl(s_A)$ measures the total length of the guards in s_A.

Logic. Given a model M and a state q in M, the $\mathbf{ATL}$ formula $\langle\langle A \rangle\rangle \gamma$ holds in M, q if there exists a general strategy s_A for agents A that ensures γ on all outcome paths starting from the epistemic class of q [Alur *et al.*, 2002; Schobbens, 2004]. Analogously, the $\mathbf{NatATL}$ formula $\langle\langle A \rangle\rangle^{\leq k} \gamma$ holds in M, q if there is a natural strategy s_A such that (i) s_A ensures γ on all outcome paths from the epistemic class of q, and (ii) $compl(s_A) \leq k$ [Jamroga *et al.*, 2019b; Jamroga *et al.*, 2019c].

```
Agent Voter1:
LOCAL: [Voter1_vote1, Voter1_vote2]
PERSISTENT: [Voter1_vote1, Voter1_vote2]
INITIAL: []
init q0
voter1vote1: q0 -> q1 [Voter1_vote1:=1]
shared[2] gv_1_Voter1[gv_1_Voter1]: q1 [Voter1_vote1==1] -> q2
voter1vote2: q0 -> q1 [Voter1_vote2:=1]
shared[2] gv_2_Voter1[gv_2_Voter1]: q1 [Voter1_vote2==1] -> q2
shared[2] ng_Voter1[ng_Voter1]: q1 -> q2
shared[2] pun_Voter1[pn_Voter1]: q2 -> q3
shared[2] npun_Voter1[pn_Voter1]: q2 -> q3
```

Figure 1: Voter specification in Asynchronous Simple Voting

4 Scenarios

We will evaluate our tool on two scalable voting scenarios.

Simple Voting. A simple scalable benchmark is provided by the Asynchronous Simple Voting scenario [Jamroga *et al.*, 2019a]. The system consists of k voters and a single coercer. The **NatSTV** code specifying the behavior of a voter is shown in Figure 1. The voter first casts their vote, then decides whether to share its value with the coercer. Finally, the voter waits for the coercer's decision to either punish or refrain from punishment. The coercer has two actions available per voter: to punish the voter or to refrain from punishment, resulting in $2k$ actions in total.

The model contains the following propositional variables: $vote_{i,j}$, indicating whether voter i has voted for candidate j; pun_i, indicating whether voter i was punished; $finish_i$, indicating whether voter i has completed the voting process and interactions with the coercer.

vVote. The second benchmark is based on a real-life voting protocol, namely the vVote system [Culnane *et al.*, 2015] that was used for the 2014 election in Victoria, Australia. We refer to [Ryan, 2010] and [Jamroga *et al.*, 2021a] for the details of the protocol and the formal model.

5 Technology

NatSTV performs *explicit-state model checking*, where the global states and transitions of the model are explicitly stored in the memory during the verification process. Users can load and parse the input specification from a text file that defines the modules, which are local automata representing the agents. The generated models and verification results are displayed in an intuitive web-based graphical interface. The verification algorithms are implemented in C++, while the GUI is developed in Typescript using the Angular framework.

NatSTV executes depth-first strategy synthesis from the initial state. During this process, it evaluates whether each new action can extend the existing partial strategy without contradicting prior decisions. Upon completion, the strategy is expressed as a set of boolean conditions involving all agent variables. The tool then optimizes this representation by eliminating redundant variables and conditions, significantly reducing the complexity of the strategy.

6 Experimental Evaluation

6.1 Configuration of the Experiments

The experiments have been conducted on a computer with 8-core AMD Ryzen 7 5700X3D running at 3.20 GHz, 64 GB

#V	Model generation	ϕ_1				ϕ_2				ϕ_3			
		General synthesis	Natural synthesis	Compl. raw	Compl. opt.	General synthesis	Natural synthesis	Compl. raw	Compl. opt.	General synthesis	Natural synthesis	Compl. raw	Compl. opt.
1	0.03	<0.01	<0.01	156	26	<0.01	<0.01	9	3	<0.01	<0.01	9	3
2	0.05	<0.01	<0.01	991	131	<0.01	<0.01	9	3	<0.01	<0.01	9	3
3	0.21	0.15	0.15	4516	512	0.01	0.04	9	3	0.02	0.03	9	3
4	5.89	5.25	5.48	18043	1831	0.02	0.02	9	3	0.04	0.05	9	3
5	254.98	memout				25.02	10.15	9	3	28.56	12.68	9	3
6	timeout	-	-	-	-	-	-	-	-	-	-	-	-

Table 1: Results for Asynchronous Simple Voting with 2 candidates

#V	Model generation	General synthesis	Natural synthesis	Compl. raw	Compl. opt.
1	0.04	<0.01	0.06	797	39
2	0.24	<0.01	0.26	2170	124
3	9.02	0.43	0.54	2105	122
4	526.16	29.55	21.83	2170	124
5	timeout	-	-	-	-

Table 2: Results for vVote with 2 candidates for ϕ_4

#V	Model generation	General synthesis	Natural synthesis	Compl. raw	Compl. opt.
1	0.04	<0.01	0.02	863	42
2	0.24	<0.01	0.04	851	38
3	9.02	0.22	0.41	851	38
4	526.16	18.64	18.81	851	38
5	timeout	-	-	-	-

Table 3: Results for vVote with 2 candidates for ϕ_5

RAM and a 64-bit Linux operating system. The algorithms were implemented in C++. All execution times are reported in seconds, with the timeout set to 2 hours.

6.2 Simple Voting

For Simple Voting, we verified the following properties:

$$\phi_1 \equiv \langle\langle c \rangle\rangle \Box((\mathit{finish}_k \wedge \mathit{vote}_{i,j}) \rightarrow \mathit{pun}_i)$$
$$\phi_2 \equiv \langle\langle v_i \rangle\rangle \Box(\neg K_c \mathit{vote}_{i,j})$$
$$\phi_3 \equiv \langle\langle v_i \rangle\rangle \Box(\mathit{finish}_i \rightarrow \mathit{vote}_{i,j} \wedge \neg K_c \mathit{vote}_{i,j}).$$

Formula ϕ_1 expresses the undesirable property of *strategic punishment*, i.e., the coercer can ensure that, whenever his interaction with the voter i concludes and the voter has disobeyed and voted for candidate j, the coercer can punish the voter for that. Formula ϕ_2 captures the desirable property of *strategic anonymity*, i.e., the voter can ensure that the coercer will never know that she has voted for candidate j. Finally, ϕ_3 denotes *strategic coercion resistance*, i.e., the voter can always vote for candidate j without the coercer knowing it. We used $j = 2$ and $i = 1$ in all experiments, except for ϕ_1 where $i = k$ was used. The experimental results, scaled by the number of voter agents, are presented in Table 1. The natural strategy found by **NatSTV** for ϕ_3 is given below:

$$\neg \mathit{vote}_{1,2} \rightarrow \mathit{vote}_2$$
$$\top \rightarrow \mathit{ng}$$

Table 1 includes the following columns: scalability parameter (the number of voter agents); model generation time; synthesis time for general strategies and natural strategies; complexity of the synthesized natural strategy without and with optimization. The results demonstrate that the natural strategy synthesis requires approximately the same time as the synthesis of general strategies. The optimization of the natural strategy, although relatively straightforward, significantly reduces its complexity.

Moreover, depending on the agent and the formula, the complexity of the natural strategy may not increase with the number of agents in the system. This occurs for voter strategies, as their strategy is unaffected by other agents, unlike the coercer, who must adapt his choices depending on with whom he currently interacts.

6.3 vVote

For vVote, we consider the following properties:

$$\phi_4 \equiv \langle\langle v_1 \rangle\rangle \Diamond(\mathit{checkWBB\_ok} \vee \mathit{checkWBB\_notok})$$
$$\phi_5 \equiv \langle\langle v_1, c \rangle\rangle \Diamond(\mathit{vote}_{1,1} \wedge K_c \mathit{vote}_{1,1}).$$

Formula ϕ_4 expresses the desirable property of *voter verifiability*, indicating that the voter can verify the correctness of her vote on the web bulletin board. Then, ϕ_5 represents the undesirable property of *strategic coercion*, stating that the coercer with the support of the voter can ensure that the voter votes for the first candidate and coercer will know about that. The experimental results, scaled by the number of voter agents, are presented in Tables 2 and 3.

The results show a similar pattern as in the Simple Voting scenario. The natural strategy synthesis requires approximately the same time as the standard strategy synthesis. The reduction of the natural strategy significantly simplifies the strategy. Moreover, the complexity of the natural strategy varies only slightly with the number of agents in the system.

7 Usage

The tool can be accessed at stv.cs-htiew.com. A video demonstration of the tool is available at jmp.sh/AiRcVLmH. Sample specifications are available at stv-docs.cs-htiew.com. The current version of **NatSTV** offers the following features:

- Generate and visualize the composition of a set of modules into a global multi-agent model;
- Provide local specifications for modules and compute the global specification as their conjunction;
- Display the verification results, including relevant truth values and the winning natural strategy (if one exists).

8 Conclusions

We introduce **NatSTV**, a novel extension in the **STV** model checking suite for the synthesis and verification of natural strategies. The experiments indicate that verifying natural strategies with **NatSTV** yields performance comparable to model checking standard strategic properties. Consequently, we obtain an important new functionality with minimal impact on the complexity and performance of the suite.

Acknowledgments

The work has been supported by NCBR Poland and FNR Luxembourg under the PolLux/FNR-CORE project SpaceVote (POLLUX-XI/14/SpaceVote/2023 and C22/IS/17232062/SpaceVote). For the purpose of open access, and in fulfilment of the grant agreement, the authors have applied CC BY 4.0 license to any Author Accepted Manuscript version arising from this submission.

References

[Alur *et al.*, 1998] R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proceedings of Computer Aided Verification (CAV)*, volume 1427 of *Lecture Notes in Computer Science*, pages 521–525. Springer, 1998.

[Alur *et al.*, 2002] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[Aruta *et al.*, 2024] Marco Aruta, Vadim Malvone, and Aniello Murano. A model checker for natural strategic ability, 2024.

[Belardinelli *et al.*, 2022] Francesco Belardinelli, Wojtek Jamroga, Vadim Malvone, Munyque Mittelmann, Aniello Murano, and Laurent Perrussel. Reasoning about human-friendly strategies in repeated keyword auctions. In *Proceedings of AAMAS*, pages 62–71. IFAAMAS, 2022.

[Berthon *et al.*, 2024] Raphaël Berthon, Joost-Pieter Katoen, Munyque Mittelmann, and Aniello Murano. Natural strategic ability in stochastic multi-agent systems. In *Proceedings of AAAI*, pages 17308–17316. AAAI Press, 2024.

[Bulling *et al.*, 2010] N. Bulling, J. Dix, and W. Jamroga. Model checking logics of strategic ability: Complexity. In M. Dastani, K. Hindriks, and J.-J. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.

[Busard *et al.*, 2014] S. Busard, C. Pecheur, H. Qu, and F. 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.

[Cermak *et al.*, 2014] P. Cermak, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *Proc. of Computer Aided Verification (CAV)*, volume 8559 of *Lecture Notes in Computer Science*, pages 525–532. Springer, 2014.

[Cermák *et al.*, 2015] Petr Cermák, Alessio Lomuscio, and Aniello Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *Proceedings of AAAI*, pages 2038–2044, 2015.

[Chen *et al.*, 2013] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISM-games: A model checker for stochastic multi-player games. In *Proceedings of Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 7795 of *Lecture Notes in Computer Science*, pages 185–191. Springer, 2013.

[Culnane *et al.*, 2015] C. Culnane, P.Y.A. Ryan, S.A. Schneider, and V. Teague. vvote: A verifiable voting system. *ACM Trans. Inf. Syst. Secur.*, 18(1):3:1–3:30, 2015.

[Ferrando and Malvone, 2024] Angelo Ferrando and Vadim Malvone. Hands-on VITAMIN: A compositional tool for model checking of multi-agent systems. In *Proceedings of the 25th Workshop "From Objects to Agents"*, volume 3735 of *CEUR Workshop Proceedings*, pages 148–160. CEUR-WS.org, 2024.

[Huang and van der Meyden, 2014] X. Huang and R. van der Meyden. Symbolic model checking epistemic strategy logic. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 1426–1432, 2014.

[Jamroga *et al.*, 2018] 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, 2018.

[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.

[Jamroga *et al.*, 2019b] Wojciech Jamroga, Vadim Malvone, and Aniello Murano. Natural strategic ability. *Artificial Intelligence*, 277, 2019.

[Jamroga *et al.*, 2019c] Wojciech Jamroga, Vadim Malvone, and Aniello Murano. Natural strategic ability under imperfect information. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*, pages 962–970. IFAAMAS, 2019.

[Jamroga *et al.*, 2020a] W. Jamroga, W. Penczek, T. Sidoruk, P. Dembiński, and A. Mazurkiewicz. Towards partial order reductions for strategic ability. *Journal of Artificial Intelligence Research*, 68:817–850, 2020.

[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*, volume 12455 of *Lecture Notes in Computer Science*, pages 129–146. Springer, 2020.

[Jamroga *et al.*, 2021a] Wojciech Jamroga, Damian Kurpiewski, and Vadim Malvone. Natural strategic abilities in voting protocols. In *Proceedings of STAST 2020*, 2021. To appear.

[Jamroga *et al.*, 2021b] 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.

[Jamroga, 2015] Wojciech Jamroga. *Logical Methods for Specification and Verification of Multi-Agent Systems*. ICS PAS Publishing House, 2015.

[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 *Proceedings of AAMAS*, pages 2812–2814. ACM, 2024.

[Kurpiewski *et al.*, 2019] 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.

[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.

[Kurpiewski *et al.*, 2022] Damian Kurpiewski, Wojciech Jamroga, Lukasz Masko, Lukasz Mikulski, Witold Pazderski, Wojciech Penczek, and Teofil Sidoruk. Verification of multi-agent properties in electronic voting: A case study. In *Advances in Modal Logic*, pages 531–556. College Publications, 2022.

[Kurpiewski *et al.*, 2024] Damian Kurpiewski, Mateusz Kaminski, Yan Kim, Lukasz Masko Witold Pazderski, Wojciech Jamroga, and Lukasz Mikulski. STV – StraTegic Verifier. code repository, 2024. https://github.com/blackbat13/stv.

[Lomuscio *et al.*, 2013] Alessio Lomuscio, Ben Strulo, Nigel G. Walker, and Peng Wu. Assume-guarantee reasoning with local specifications. *Int. J. Found. Comput. Sci.*, 24(4):419–444, 2013.

[Lomuscio *et al.*, 2017] A. Lomuscio, H. Qu, and F. 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.

[Mogavero *et al.*, 2014] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic*, 15(4):1–42, 2014.

[Ryan, 2010] P.Y.A. Ryan. The computer ate my vote. In *Formal Methods: State of the Art and New Directions*, pages 147–184. Springer, 2010.

[Schobbens, 2004] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.