ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA241035

STV+FLY: On-the-Fly Model Checking of Strategic Ability in Multi-Agent Systems

Damian Kurpiewski[a,b], **Mateusz Kamiński**[a,b] **and Wojciech Jamroga**[a,c]

[a]Institute of Computer Science, Polish Academy of Sciences
[b]Faculty of Mathematics and Computer Science, Nicolaus Copernicus University in Toruń
[c]Interdisciplinary Centre for Security, Reliability and Trust, SnT, University of Luxembourg
ORCID (Damian Kurpiewski): https://orcid.org/0000-0002-9427-2909, ORCID (Mateusz Kamiński): https://orcid.org/0009-0003-5617-8959, ORCID (Wojciech Jamroga): https://orcid.org/0000-0001-6340-8845

Abstract. In this paper, we present a substantially enhanced version of our software tool **STV** (STrategic Verifier), dedicated to strategy synthesis and model checking of strategic abilities in multi-agent systems. The new extension, called **STV+FLY**, incorporates an advanced strategy synthesis algorithm that enables model checking with on-the-fly generation of the global model. This innovative approach allows for the verification of some strategic properties without generating the entire global state space, thus avoiding an important bottleneck and significantly improving the efficiency.

1 Introduction

Alternating-time temporal logic **ATL**/**ATL*** [1, 37] is a widely recognized framework for reasoning about the *strategic abilities* of agents. **ATL** enables to address the potential behaviors and outcomes of individual agents and their coalitions. Over the past two decades, substantial effort has been devoted to developing algorithms and tools for formal verification of **ATL** specifications [3, 37, 12, 34, 2, 13, 9, 17, 10, 32, 11, 5, 4, 21, 29]. However, the computational complexity of model checking for strategic abilities, especially under imperfect information, presents significant challenges [8, 15].

To address the challenges, our group at the Polish Academy of Sciences has developed a number of techniques for *incomplete* and *approximate verification* of **ATL** for agents with imperfect information [21, 28, 33, 18] as well as suitable *model reductions* [23, 6, 7]. Most of those have been implemented in our experimental open-source model checker **STV** (STrategic Verifier) [27, 29], and shown promising performance on toy models. To test their merits in a more realistic context, we conducted an extensive case study [30], verifying properties of the Selene protocol for secure voting [36]. We used an "all-out" approach, employing (in various combinations) fixpoint approximation, brute-force depth-first strategy synthesis, DFS synthesis with elimination of dominated partial strategies, partial-order reduction, and two different approaches to distributed verification. The results in [30] showed that, in realistic scenarios, model generation is often the primary bottleneck. Whenever the verification failed, it was because there was not enough time or memory to generate the global state/transition space in the first place. Since the size of the model grows exponentially with every added agent, so do the associated time and memory requirements.

In this work, we observe that in some cases the generation of the whole global model is not necessary. This happens especially when a winning state can be reached in few steps, or the search algorithm is "lucky," and hits the right choices first. Based on that observation, we introduce an innovative extension of **STV**, which incorporates a novel approach based on *depth-first strategy synthesis* with *on-the-fly model generation*. The method enables to verify the specified properties without the prerequisite of generating the entire model in advance. Instead, the model is constructed incrementally during the verification process, with only those parts being generated, that are necessary to do the next verification step.

2 Application Domain

STV+FLY addresses formal verification of Multi-Agent Systems (MAS), which is a notoriously complex and sophisticated problem [14]. It is generally accepted that most relevant system requirements in MAS refer to the strategic ability (or inability) of agents and their groups. For instance, the **ATL*** formula $\langle\langle taxi \rangle\rangle \mathrm{G} \neg \mathsf{fatality}$ says that an autonomous cab can operate in a manner that ensures no fatalities. Similarly, $\langle\langle taxi, passg \rangle\rangle \mathrm{F}\, \mathsf{destination}$ denotes that the cab and passenger can collaboratively reach the destination, regardless of the actions from other agents. An even richer group of requirements can be specified using a mix of strategic and knowledge operators, which is especially relevant in information security (e.g., anonymity, privacy, and successful information exchange) and the analysis of voting procedures (e.g., receipt-freeness, coercion resistance, and voter-verifiability [35, 38]).

Unfortunately, verification of strategic abilities (with and without knowledge) is challenging both theoretically and practically. In particular, practical verification of such properties remains challenging due to exponential explosion of the state space and transition space, and *at least exponential* explosion of the strategy space (on top of the former explosion!). This has been demonstrated in various case studies, e.g., in [19, 22, 24, 30].

STV, and its new extension **STV+FLY**, provide a user-friendly environment tailored for the analysis of such requirements. They feature a graphical user interface (GUI) and a flexible model specification language, enhancing the accessibility and usability of the model checker. Additionally, the tool serves significant pedagogical value, offering an intuitive platform for introducing the topic of strategic reasoning and verification of strategic logics. Previous versions of

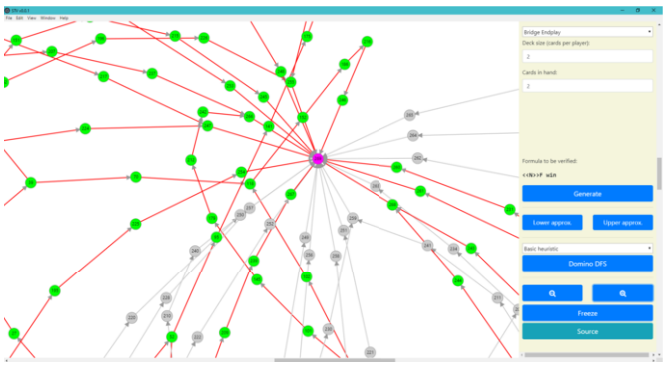

Figure 1. Screenshot of **STV**

STV have been successfully employed in educational settings, including tutorials and graduate courses at major AI conferences and summer schools, such as IJCAI, PRIMA, and ESSAI.

3 Formal Background

Model checking asks if a given model of the system satisfies a given property. Thus, the input typically consists of the model (or its symbolic representation) and a logical formula expressing the property. For the former, we use modular representations based on Asynchronous Multi-Agent Systems [20, 25] and a flexible model specification language [29], in which each agent is specified using separate local model. For the latter, we employ formulas of **ATL**, interpreted over memoryless strategies with imperfect information [37].

Modules. The main part of the input is given by a set of asynchronous modules [31, 23], where local states (states of the agent local model) are labelled with valuations of state variables. The transitions are valuations of input variables controlled by the other modules. The global model of the MAS is defined by the asynchronous product of its modules.

Strategies. A strategy is a conditional plan that specifies what the agent(s) are going to do in every possible situation [3, 37]. Here, we consider the case of *imperfect information memoryless strategies*, represented by functions from the agent's local states to its available actions. The *outcome* of a strategy from state q consists of all the infinite paths starting from q and consistent with the strategy.

Logic. Given a model M and a state q in the model, the formula $\langle\langle A\rangle\rangle\varphi$ holds in M, q iff there exists a strategy for agents A that makes φ true on all the outcome paths starting from any state indistinguishable from q [3, 37]. The semantics of coalitional abilities is analogous, for joint strategies of coalitions. Following this concept, the formula $\langle\langle S\rangle\rangle\varphi$ holds in (M, q) iff there exists a joint strategy for coalition S (i.e., a set of strategies, one for every agent in S) that makes φ true on all the outcome paths starting from q. Moreover, $K_a\varphi$ (agent a *knows* φ) holds in M, q iff φ is true in every state q' indistinguishable from q for a [16].

4 Technology

STV+FLY does *explicit-state model checking*. That is, the global states and transitions of the model are represented explicitly in the memory of the verification process. The user can load and parse the input specification from a text file that defines the modules, i.e., local automata representing the agents. The generated models and the verification results are visualised in an intuitive web-based graphical interface. The verification algorithms are implemented in C++, and the GUI in Typescript, using the Angular framework.

5 Benchmark

As a benchmark for testing our tool we use Selene voting protocol, which was already used in [30] to assess the performance of our previous verification techniques.

Selene [36] is an electronic voting protocol designed to ensure *voter verifiability* and *coercion resistance*. This protocol equips the voter with evidence to confirm that her vote has been accurately registered. Simultaneously, it allows her to deceive a coercer by providing falsified evidence, thereby protecting her true voting intentions.

The protocol operates in several stages:

1. **Setup**: Prior to the election, the Election Authority (EA) prepares the system by generating keys for vote encryption and decryption and assigning a unique vote tracker to each voter. These trackers are encrypted and shuffled to anonymize the link between voters and their respective trackers, then published on the Web Bulletin Board (WBB).
2. **Voting**: Voters encrypt and sign their ballots before submitting them to the system. The system processes these ballots through several intermediate steps, culminating in the publication of pairs $(Vote_v, tr_v)$ on the WBB for each voter. At this point, voter identities remain anonymous, and the decrypted ballots allow for a public tally and audit.
3. **Disclosure**: In the final stage, voters receive their trackers via an independent channel (e.g., SMS). Voters not subjected to coercion request a special term (α_v) to retrieve their actual tracker. Conversely, coerced voters who are compelled to vote in a specific manner communicate the demanded vote to the election server. In response, they receive a counterfeit term (α'_v) which they can present to the coercer. This term reveals a tracker for a vote that appears to align with the coercer's demands.

Selene employs the ElGamal encryption scheme, leveraging its multiplicative homomorphism and non-interactive zero-knowledge proofs of knowledge for all data transformations displayed on the WBB. In this study, we abstract from the cryptographic details, and focus on the strategic interaction between the involved parties.

Our model includes four distinct agent templates: the Election Authority (EA), the Coercer, the standard Voter, and the Coerced Voter. The EA is responsible for generating and distributing trackers and managing the Web Bulletin Board (WBB), which is accessible to both voters and coercers.

The procedure begins with the Coerced Voter interacting with the Coercer, who may demand a vote for a specific candidate. Following this, the voting procedure commences, synchronized with the EA. During this phase, the voter generates her commitment, completes her ballot, encrypts it, and submits it to the EA. Then, she waits for the votes to be published on the WBB, which is a process overseen by the EA.

After vote publication, the Coerced Voter can fabricate an alpha-term and fake tracker or await the real tracker. The Voter then checks the WBB, confirms her vote, and interacts with the Coercer. Depending on the scenario, she may present either the false or the real tracker to the Coercer, who will then decide whether to penalize her.

Additionally, the voter has the option to *revote*, casting multiple votes as a strategy to mitigate domestic coercion by family members. In case of revoting, only the latest ballot counts.

#A	Standard					On-the-fly		Res
	States	Old		New		States	Verif	
		Gen	Verif	Gen	Verif			
4	3.85e4	10	<1	3	<1	2.91e3	<1	True
5	2.19e6	520	<1	179	<1	1.47e5	1	True
6	8.12e7	10252	<1	2642	<1	1.10e6	14	True
7	timeout					9.60e6	406	True
8	timeout							

Table 1. Results for ϕ_1 with 3 candidates and 3 revotes

#A	Standard					On-the-fly		Res
	States	Old		New		States	Verif	
		Gen	Verif	Gen	Verif			
4	1.09e4	1	<1	<1	<1	19	<1	True
5	5.44e5	133	<1	37	<1	83	<1	True
6	3.34e7	2412	3	528	2	731	<1	True
7	timeout					5.43e3	32	True
8	timeout					1.96e4	9529	True
9	timeout							

Table 2. Results for ϕ_2 with 2 candidates and 3 revotes

6 Experimental Evaluation

We have assessed the performance of our new algorithm using the Selene voting protocol as a testbed. Our evaluation comprised two sets of tests, each focusing on specific properties defined by the following formulas of **ATLK**:

$$\phi_1 \equiv \langle\langle C\rangle\rangle \mathrm{G}\big((\mathsf{finish}_1 \wedge \mathsf{revote} = 2 \wedge \mathsf{voted}_1 = 1) \rightarrow K_C \mathsf{voted}_1 = 1\big)$$

$$\phi_2 \equiv \langle\langle V_1\rangle\rangle \mathrm{F} \bigvee_i (\mathsf{voted}_i = 1)$$

The first formula, ϕ_1, is the same as the one used for experiments in [30].[1] It expresses that, if the coercer concludes his interaction with the coerced voter who then votes for her intended candidate[2] without doing the final revote, then the coercer will know how the voter has voted. Like in [30], we applied this formula to a family of Selene models with 3 candidates and 3 revoting phases. The number of agents #A was used as the scalability parameter. The results of the experiments are presented in Table 1. Note that in this particular setting the system always includes 1 Coercer, 1 Election Authority and 1 Coerced Voter, plus an arbitrary number of ordinary Voters (at least 1), although number of agents could be easily modified.

The table provides a comparative analysis of the results obtained using the "standard" verification approach (generate the full global model first, verify next), versus the new on-the-fly verification method implemented in **STV+FLY**. We report the number of generated states, the model generation time, and the verification time. All times are given in seconds. The timeout was set to $3h$. The test platform was a server equipped with ninety-six 2.40 GHz Intel Xeon Platinum 8260 CPUs, 991 GB RAM, and 64-bit Linux. For "standard" verification, we include two sets of results: one obtained by the old version of **STV** used in [30], with model generation implemented in Python, and the other using the most recent version of **STV** [26], done fully in C++.

The second formula, ϕ_2, says that the voter can ensure at least one vote for the selected candidate to appear in the tally. The tests were conducted using Selene models with 2 candidates and 3 revoting phases. The outcomes are presented in Table 2.

The results of the experiments indicate that on-the-fly model checking allows for verification of much larger models. The new approach enabled to analyze Selene instances with one more voter (for ϕ_1) and two more voters (in case of ϕ_2). Even more interestingly, the on-the-fly algorithm generated only 1% of the state space (for ϕ_1 and $\#A = 6$ agents), and as little as 0.001% in case of ϕ_2 and $\#A = 6$. Finally, note that the size of the full model grows geometrically, roughly speaking 50 times with each added voter. Thus, for ϕ_2, **STV+FLY** was able to verify models that were approximately *1000 times larger* than the ones amenable to standard explicit-state model checking.

7 Usage

The tool is available at http://stv.cs-htiew.com/. The video demonstration of the tool is available at https://www.youtube.com/watch?v=eDlTVhTbnhk. Example specifications can be found at http://stv-docs.cs-htiew.com/. The current version of **STV** allows to: (i) generate and display the composition of a set of modules into the model of a multi-agent system; (ii) provide local specifications for modules, and compute the global specification as their conjunction; (iii) verify an **ATL**, **ATLK** and/or **ATLH** reachability or safety formula with knowledge and/or uncertainty operators (nested strategic operators are not allowed); (iv) finally, display the verification result including the relevant truth values.

8 Conclusions

We present **STV+FLY**, which offers a substantial revision and extension of the **STV** model checker. Compared to the previous version, **STV+FLY** features a new implementation of the verification algorithm in C++. More importantly, it significantly changes the way the strategy synthesis is conducted. Instead of generating the (excessively huge) global model first, and running the verification algorithm afterwards, the model is generated incrementally as the immediate need dictates.

The experiments show significant gains in performance. In particular, on-the-fly verification of coercion resistance and voter enfranchisement in a well-known voting protocol succeeded for problems with 1000 larger state/transition spaces, compared to the traditional pipeline of "first generate, then verify." This shows that, indeed, only a small part of the global model needs to be explored in many practical scenarios.

Acknowledgements

We thank Łukasz Mikulski for his comments and help in preparing the benchmark and formulas for evaluation. 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), by NCN Poland under the CHIST-ERA grant SAI (CHIST-ERA-19-XAI-010, 2020/02/Y/ST6/00064), by FNR Luxembourg under the CORE project PABLO (C21/IS/16326754/PABLO), and by the CNRS IEA project MoSART. For the purpose of open access, and in fulfilment of the obligations arising from the grant agreement, the authors have applied CC BY 4.0 license to any Author Accepted Manuscript version arising from this submission.

[1] With the small difference that, at that time, **STV** did not admit knowledge operators, so epistemic formulas had to be emulated by additional atomic propositions.

[2] We assume without loss of generality that it is the candidate number 1.

References

[1] R. Alur, T. A. Henzinger, and O. 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.

[2] 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.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002. doi: 10.1145/585265.585270.

[4] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of broadcasting multi-agent systems against an epistemic strategy logic. In *Proceedings of IJCAI*, pages 91–97, 2017.

[5] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multi-agent systems with imperfect information and public actions. In *Proceedings of AAMAS*, pages 1268–1276, 2017.

[6] F. Belardinelli, R. Condurache, C. Dima, W. Jamroga, and M. Knapik. Bisimulations for verifying strategic abilities with an application to the ThreeBallot voting protocol. *Information and Computation*, 276: 104552, 2021. doi: 10.1016/j.ic.2020.104552.

[7] F. Belardinelli, A. Ferrando, W. Jamroga, V. Malvone, and A. Murano. Scalable verification of strategy logic through three-valued abstraction. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI*, pages 46–54. ijcai.org, 2023. doi: 10.24963/IJCAI.2023/6.

[8] 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.

[9] 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. ISBN 978-3-319-11736-2. doi: 10.1007/978-3-319-11737-9_3.

[10] 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.

[11] P. Cermák, A. Lomuscio, and A. Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *Proceedings of AAAI*, pages 2038–2044, 2015.

[12] K. Chatterjee, T. Henzinger, and N. Piterman. Strategy Logic. *Information and Computation*, 208(6):677–693, 2010.

[13] 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.

[14] M. Dastani, K. Hindriks, and J. Meyer, editors. *Specification and Verification of Multi-Agent Systems*. Springer, 2010.

[15] C. Dima and F. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[16] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[17] 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.

[18] W. Jamroga and D. Kurpiewski. Pretty good strategies and where to find them. In *Proceedings of EUMAS*, volume 14282 of *Lecture Notes in Computer Science*, pages 363–380. Springer, 2023. doi: 10.1007/978-3-031-43264-4_23.

[19] 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.

[20] 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, 2018.

[21] W. Jamroga, M. Knapik, D. Kurpiewski, and Ł. Mikulski. Approximate verification of strategic abilities under imperfect information. *Artificial Intelligence*, 277, 2019. doi: 10.1016/j.artint.2019.103172.

[22] W. Jamroga, Y. Kim, D. Kurpiewski, and P. 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. doi: 10.1007/978-3-030-60347-2_9.

[23] 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. doi: 10.1613/jair.1.11936.

[24] W. Jamroga, D. Kurpiewski, and V. Malvone. Natural strategic abilities in voting protocols. In *Proceedings of STAST 2020*, 2021. To appear.

[25] W. Jamroga, W. Penczek, and T. Sidoruk. Strategic abilities of asynchronous agents: Semantic side effects and how to tame them. In *Proceedings of KR 2021*, pages 368–378, 2021.

[26] M. Kaminski, D. Kurpiewski, and W. Jamroga. STV+KH: towards practical verification of strategic ability for knowledge and information flow. In *Proceedings of AAMAS*, pages 2812–2814. ACM, 2024. doi: 10.5555/3635637.3663296.

[27] D. Kurpiewski, W. Jamroga, and M. 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.

[28] D. Kurpiewski, M. Knapik, and W. 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.

[29] D. Kurpiewski, W. Pazderski, W. Jamroga, and Y. Kim. STV+Reductions: Towards practical verification of strategic ability using model reductions. In *Proceedings of AAMAS*, pages 1770–1772. ACM, 2021.

[30] D. Kurpiewski, W. Jamroga, L. Masko, L. Mikulski, W. Pazderski, W. Penczek, and T. Sidoruk. Verification of multi-agent properties in electronic voting: A case study. In *Advances in Modal Logic*, pages 531–556. College Publications, 2022.

[31] A. Lomuscio, B. Strulo, N. G. Walker, and P. Wu. Assume-guarantee reasoning with local specifications. *Int. J. Found. Comput. Sci.*, 24(4): 419–444, 2013. doi: 10.1142/S0129054113500123.

[32] 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. doi: 10.1007/s10009-015-0378-x.

[33] L. Mikulski, W. Jamroga, and D. Kurpiewski. Assume-guarantee verification of strategic ability. In *Proceedings of PRIMA*, volume 13753 of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2022. doi: 10.1007/978-3-031-21203-1_11.

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

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

[36] P. Ryan, P. 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.

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

[38] M. Tabatabaei, W. Jamroga, and P. Y. A. Ryan. Expressing receipt-freeness and coercion-resistance in logics of strategic ability: Preliminary attempt. In *Proceedings of the 1st International Workshop on AI for Privacy and Security, PrAISe@ECAI 2016*, pages 1:1–1:8. ACM, 2016. doi: 10.1145/2970030.2970039.