FoMSESS Jahrestagung 2021

Extended Abstracts

Die Fachgruppe FoMSESS[1] im GI-Fachbereich Sicherheit beschäftigt sich mit der Anwendung von Formalen Methoden und Software Engineering in der Entwicklung sicherer Systeme.

In ihren Jahrestreffen bietet die Fachgruppe die Möglichkeit, über aktuelle Forschungsarbeiten zu berichten und zu diskutieren und sich mit Gleichgesinnten zu vernetzen.

Nach dem ersten Online-Jahrestreffen im Jahr 2020 wurde auch das Jahrestreffen 2021 online durchgeführt. Auch diesmal gelang es, zwei Nachmittage mit interessanten Vorträgen und lebhaften Diskussionen zu füllen. Die Vortragenden bekamen auch dieses Jahr die Möglichkeit, Extended Abstracts ihrer Beiträge zu verfassen, um diese auf der FoMSESS-Webseite zu veröffentlichen. Das Ergebnis sehen Sie gerade vor sich.

Viel Spaß beim Lesen!

Zoltan Mann, November 2021

[1] https://fg-fomsess.gi.de

Public Confidence and Internet Voting

Carsten Schürmann
Center For Information Security and Trust
IT University of Copenhagen
Denmark
carsten@itu.dk

October 28, 2021

A voting system should not merely report the outcome: it should also provide sufficient evidence to convince reasonable observers that the reported outcome is correct. Many systems, notably paperless DRE machines still in use in US elections, fail this requirement, and for Internet Voting systems it is completely unclear how to implement the requirementat all. Rivest and Wack proposed the principle of software independence (SI) as a guiding principle and requirement for voting systems [2]. In essence, a voting system is SI if its reliance on software is "tamper-evident", that is, if there is a way to detect that material changes were made to the software without inspecting that software. This important notion has so far been formulated only informally.

In my talk, I provide more formal mathematical definitions of SI. This exposes some subtleties and gaps in the original definition, among them: what elements of a system must be trusted for an election or system to be SI, how to formalize "detection" of a change to an election outcome, the fact that SI is with respect to a set of detection mechanisms (which must be legal and practical), the need to limit false alarms, and how SI applies when the social choice function is not deterministic.

This talk is based on joint work with Wojciech Jamroga, Peter Y.A. Ryan, Steve Schneider and Philip B. Stark [1].

References

[1] Wojciech Jamroga, Peter Y.A. Ryan, Steve Schneider, Carsten Schürmann, and Philip B. Stark. A declaration of software independence. In *Festschrift in honor of Joshua Guttman*, 2021. to appear.

[2] R.L. Rivest and J.P. Wack. On the notion of "software independence" in voting systems (draft version of July 28, 2006). Technical report, Information Technology Laboratory, National Institute of Standards and Technology, 2006.

Verified Synthesis and Test of Safety Supervisors in Human-Robot Collaboration – Extended Abstract

Mario Gleirscher, Universität Bremen

October 20, 2021

Abstract

This extended abstract summarises our ongoing research into a correct-by-construction approach to designing specific discrete-event controllers called *safety supervisors*. The approach has been demonstrated in a human-robot collaboration setting but has the potential to be used in other safety-critical domains where regulations require safety-related devices to be certified before use.

The increased use of artificial intelligence (AI) technologies, particularly machine learning algorithms, in safety-critical automation (i.e., autonomous aerial, ground, and marine vehicles; robotic transportation, logistics, and manufacturing; medical, agricultural, building, and home automation) has vast implications on the modernisation of automatic safety measures used across automation domains. The operation of systems in such domains is, for many reasons, highly regulated, and certification requires the assurance of a conglomerate of critical system properties called *operational safety*. These properties include functional safety in terms of fail-safe properties to be shown of the involved electronic and software-based control systems. Traditionally, the assurance of functional safety focuses on arguing that the nominal behaviour of a system fulfils its specification and, more importantly, potentially dangerous deviations from this specification (i.e., hazardous failures) are covered by safety functions (e.g. safety monitoring, control, and shutdown devices). Technological progress coincides with a regular revision of these functions. Furthermore, increased automation has led to an extension of this assurance strategy, namely to show that the specification of nominal behaviour actually reflects operational safety as understood by domain experts and risk analysts. In domains, such as advanced driver assistance, this aspect of assurance is known as safety of the intended function (SOTIF) and standardised in ISO/PAS 21448 (2019).

Human-robot collaboration is a domain where controllers of robots and machines have to guarantee safety under uncertainty, for example, a high probability of collision freedom, a reduction of impact frequency, or impact force minimisation, while dealing with a complex uncontrollable environment including humans. Modern human-robot collaboration strives for finer-quality interaction between humans and machines as well as more autonomy of the involved robots and other machines to increase the productivity, the complexity of manageable tasks, and the ergonomic division of labour. This development calls for a paradigm shift from the traditional safety monitoring and shutdown philosophy towards a more holistic notion of safety supervision to be delivered by functions of a machine's control system. We speak of these functions as *safety supervisors*. Supervisors can include complex logic—sensing, computation, and actuation, even using AI—to ensure safe optimal collaboration between humans and machines.

Challenging for engineers is the reoccurring question of how one can derive a controller—by design, one that includes a supervisor—such that the nominal or failure behaviour of the controlled process fulfils the requirement of operational safety for that process? Particularly, how can one integrate application control with safety supervision? And, for validation and verification, how does one represent these requirements, that is, the assumptions about the environment as well as the guarantees of the controller?

Among the numerous ways to do this, we have chosen a stochastic action language and a probabilistic temporal logic to model the process, to specify the requirements, and to describe the controller designs in order to reason about supervisor behaviours in a clean and well-understood way through transition systems. Moreover, based on this framework, we propose a *correct-by-construction approach* including (a) early supervisor validation, (b) verified synthesis of abstract (platform-independent) supervisors, and (c) the generation and verification of concrete (platform-specific) supervisor implementations.

Stage (a) takes advantage of task and risk modelling (Gleirscher, Calinescu, and Woodcock 2021) and applies stochastic model checking for model correctness proofs based on operational safety properties (Gleirscher, Calinescu, Douthwaite, et al. 2021). Stage (b) uses Markov decision process policy synthesis (Gleirscher and Calinescu 2020) for the extraction of an optimal abstract supervisor from a synthesised policy. Stage (c) utilises recent advances in complete conformance testing (Huang et al. 2016) to verify the translation of abstract supervisors into executable supervisor implementations.

One of the advantages of the outlined approach (Gleirscher and Peleska 2021) is that the stages (b) and (c) exhibit a high degree of automation. Hence, during a SOTIF-style assurance, domain experts and verification engineers can focus on the model and requirements validation part while receiving support through model checking. They can, for example, focus on translating regulatory requirements into a complete and non-vacuous set of properties to be checked of the process model. This way, validation rules out common-cause mistakes, a typical weakness of correct-by-construction software engineering approaches. Furthermore, the separation of synthesis into two stages (i.e., (b), automata synthesis and, (c), code generation) allows the abstract supervisor to be kept simple and facilitates efficient synthesis in more complex applications. This separation also provides freedom for code generation with manually crafted (interface) refinements needed for the integration into potentially several desired control platforms. Error possibilities coming along with this freedom can then be uncovered by complete conformance testing (Huang et al. 2016), given the assumptions for this form of testing hold (i.e., abstract and concrete controllers have the same number of control states; inputs to the concrete controller form equivalence classes modulo an isomorphism to the inputs of the abstract controller). A successfully tested conformance then *proves*—by establishing observational equivalence between abstract and concrete supervisors—that the translation from a valid model has resulted in a correct supervisor implementation.

In our previous research, we have demonstrated a preliminary version of this approach in the context of a case study on human-robot collaboration in manufacturing (Gleirscher, Calinescu, Douthwaite, et al. 2021; Gleirscher and Peleska 2021). We are convinced that the outlined approach is generic enough to be useful in many other AI-driven and safety-critical application domains as well.

References

Huang, Wen-ling and Jan Peleska (Nov. 2016). "Complete model-based equivalence class testing for nondeterministic systems". In: *Form Asp Comput* 29.2, pp. 335–364. DOI: 10.1007/s00165-016-0402-2.

ISO/PAS 21448 (2019). *Road vehicles – Safety of the intended functionality (SOTIF)*. Standard. International Standards Organisation. URL: https://www.iso.org/standard/70939.html.

Gleirscher, Mario and Radu Calinescu (2020). "Safety Controller Synthesis for Collaborative Robots". In: *Engineering of Complex Computer Systems (ICECCS), 25th Int. Conf., Singapore*. Ed. by Yi Li and Alan Liew. ACM, pp. 83–92. DOI: 10.1109/ICECCS51672.2020.00017. arXiv: 2007.03340 [cs.RO cs.SE cs.SY eess.SY].

Gleirscher, Mario, Radu Calinescu, James Douthwaite, et al. (2021). *Verified Synthesis of Optimal Safety Controllers for Human-Robot Collaboration*. Working paper arXiv:abs/2106.06604. U York, U Sheffield, and U Bremen. arXiv: 2106.06604 [cs.RO cs.SE].

Gleirscher, Mario, Radu Calinescu, and Jim Woodcock (2021). "Risk Structures: A Design Algebra for Risk-Aware Machines". In: *Form Asp Comput* 33., pp. 763–802. DOI: 10.1007/s00165-021-00545-4. arXiv: 1904.10386 [cs.SE].

Gleirscher, Mario and Jan Peleska (2021). "Complete Test of Synthesised Safety Supervisors for Robots and Autonomous Systems". In: *Formal Methods for Autonomous Systems (FMAS), 3rd Workshop*. Ed. by Matt Luckuck and Marie Farrell. Vol. 348. EPTCS, pp. 101–109. DOI: 10.4204/EPTCS.348.7.

Tally-Hiding Verifiable E-Voting

Ralf Küsters
University of Stuttgart
ralf.kuesters@sec.uni-stuttgart.de

Abstract. Modern electronic voting systems (e-voting systems) are designed to provide not only vote privacy but also (end-to-end) verifiability. Several verifiable e-voting systems have been proposed in the literature, with Helios being one of the most prominent ones. Almost all such systems, however, reveal not just the voting result but also the full tally, consisting of the exact number of votes per candidate or even all single votes. There are several situations where this is undesirable. For example, in elections with only a few voters (e.g., boardroom or jury votings), revealing the complete tally leads to a low privacy level, possibly deterring voters from voting for their actual preference. In other cases, revealing the complete tally might unnecessarily embarrass some candidates. Often, the voting result merely consists of a single winner or a ranking of candidates, so revealing only this information but not the complete tally is sufficient. This property is called tally-hiding and it offers completely new options for e-voting.
In this talk, I will cover some of the main security properties modern e-voting system should satisfy and discuss their sometimes surprising relationships. I will also present the first provably secure verifiable and tally-hiding e-voting system, called Ordinios [1,2]. Finally, I will briefly discuss the formal analysis of such and other e-voting systems.

This work was in part funded by the Deutsche Forschungsgemeinschaft (DFG) KU 1434/11-1 and the Center for Integrated Quantum Science and Technology (IQST).

References

1. R. Küsters, J. Liedtke, J. Müller, D. Rausch, and A. Vogt, "Ordinos: A Verifiable Tally-Hiding E-Voting System," in *EuroS&P 2020*. IEEE, 2020, pp. 216–235.
2. F. Hertel, N. Huber, J. Kittelberger, R. Kuesters, J. Liedtke, and D. Rausch, "Extending the Tally-Hiding Ordinos System: Implementations for Borda, Hare-Niemeyer, Condorcet, and Instant-Runoff Voting," in *Electronic Voting - Sixth International Joint Conference on Electronic Voting, 5 – 8 October 2021, Online, Proceedings*, 2021.

Hyperion: An Enhanced Version of the Selene End-to-End Verifiable Voting Scheme

Peter Y. A. Ryan[1], Simon Rastikian[2], and Peter B. Rønne[1]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.ryan@uni.lu, peter.roenne@gmail.com
[2] École Normale Supérieure, Paris, France
simon.rastikian@ens.fr

"Everything should be made as simple as possible, but no simpler."

— A. Einstein

Introduction We present a novel, end-to-end verifiable scheme, Hyperion, inspired by the Selene scheme [1], which similarly provides highly transparent verification: voters check their vote directly in plaintext in the tally. It has a number of advantages including eliminating the tracker collision threats in Selene, indeed our construction does not need trackers for verification. The new scheme should give voters a greater sense of privacy.

In the original Selene, vote/tracker pairs are revealed in the tally on the Bulletin Board. Voters are later notified of their tracker: by providing them with a private "alpha" term which along with their private, trapdoor key, opens their commitment to reveal their tracker. As long as the trackers remain private and deniable ballot privacy is preserved. However, some voters, understandably, find the public posting of the trackers alongside the votes troubling. Furthermore, Selene suffers from the possibility of a coercer claiming that the alternate tracker proffered by a coerced voter is their own.

Hyperion, by contrast, does not publicly reveal trackers, indeed, we can do away entirely with trackers. Instead, the voter identifies her vote in the tally by identifying the commitment which, along with her "alpha" term and her trapdoor key opens to a constant, e.g. the identity 1. This is rather like identifying your house by finding the door that opens to your key. This is still deniable, but the mechanism is now different: a coerced voter identifies a commitment paired with the coercer's required vote and, if necessary, computes using her trapdoor secret key, the fake alpha term that opens this to 1.

At first glance this seems to counter the tracker collision threat, but we have just shifted the problem: now the coercer might claim that the commitment the voter points to is theirs. To counter this we propose a further innovation: each voter gets an individual view of Bulletin Board BB. Each view is veifiably derived from the BB with its own, independent shuffling. Thus the rows and betas appear in a different form and order for each voter, so even a shoulder-surfing coercer cannot identify his beta in the voter's view.

The Setup Signing keys for the voters, $\mathrm{PK_i}$ are published on the master bulletin board, one row per voter. For voter V_i trapdoor keys, x_i and $h_i := g^{x_i}$, are generated by the voter's app and h_i is registered along with the casting of the vote, along with suitable ZK proofs of knowledge of x_i.

Voting Voting is much as in Selene: V_i sends an encryption of her vote with the associated plaintext awareness and well-formedness proofs, and her public trapdoor key h_i along with ZK proofs of knowledge of x_i. We denote the concatenation of these proofs by Π_i. This is signed and posted to the master BB against $\mathrm{PK_i}$:

$$\mathrm{PK_i},\ \mathrm{Sign_i}(\{\mathrm{Vote_i}\},\ h_i,\ \Pi_i)$$

Tellers now generate the analogues of the alpha and beta terms of Selene: each trapdoor public key h_i is raised to a fresh, secret random r_i. The corresponding (pre-)alpha term g^{r_i} is kept secret for the moment by the tellers:

$$\mathrm{PK_i},\ \mathrm{Sign_i}(\{\mathrm{Vote_i}\},\ h_i,\ \Pi_i),\ h_i^{r_i}$$

Tallying On the BB, ballots with valid signatures and proofs are identified and for these we extract the encrypted votes and beta commitments:

$$(\{\mathrm{Vote_j}\},\ h_j^{r_j})$$

These are now put through verifiable, hybrid, parallel mixes: the vote terms are subjected to a conventional re-encryption mix, but the commitment terms are subjected to an "exponentiation" mix: all raised to a common, secret exponent s. Such mixes can be implemented using Verificatum in suitable modes, [2]. Finally, the votes are verifiable decrypted outputting:

$$(\mathrm{Vote_j},\ h_j^{r_j \cdot s})$$

The rows are now sorted to group votes for the same candidate together. For V_i we now create an individual view by applying a further parallel mix, but here we just permute and re-randomise within the candidate groups. For each voter there will be a different, independent common exponent s_i for the exponentiation mix of the commitment terms:

$$(\mathrm{Vote_j},\ (h_j^{r_j})^{s \cdot s_j})$$

The mix tellers keep secret for now the "alpha" terms: $\alpha_i = g^{r_i \cdot s \cdot s_i}$.

Note that the commitment terms are not opened or decrypted, thus the plaintext votes appear paired with cryptographic blobs.

At notification time, α_i is sent to V_i who raises this to x_i and finds the match among the beta terms in her view, so identifying her vote. In the event of coercion, V_i identifies a row containing the coercer's required vote and computes the alpha which, when raised to x_i, matches the beta in this row.

Discussion We note that the individual views are only necessary if we want to fully counter the tracker/commitment collision problem and render the scheme fully coercion resistant. Even without the individual views the scheme provides the same guarantees as Selene (e.g. receipt-freeness and coercion mitigation). It is significantly simpler and provides a greater sense of privacy than Selene.

We can in fact retain trackers in our construction, and this may inspire a greater sense of assurance in the verification. In this case, we assign trackers in the setup phase, as in Selene. Now a coerced voter computes a fake alpha that opens the alternative beta to **her own** tracker, i.e., no need to identify a fake tracker. Furthermore, the association of trackers with voters can now be made public! This allows universal verification that all the trackers are distinct. Note that in this construction each voter only see their own tracker in their own view. Nowhere are all the trackers displayed alongside the votes, so the privacy concerns of Selene do not arise.

Conclusions We have outlined Hyperion, a new scheme, with three variants, that provides voters with a similarly direct, intuitive way to verify that their votes are correctly included in the tally. It is conceptually much simpler than Selene and avoids the tracker collision threats, indeed we can do away with trackers altogether. Furthermore, voters should feel more comfortable with this scheme as it does not involve the public posting of tracker/vote pairs.

Achieving full coercion resistance comes at the cost of introducing individual views for each voter, but this should give voters a greater sense of privacy. The individual boards may be better suited for smaller elections, where the collision threat is more troublesome. The construction without individual boards, may be useful for large elections where the collision problem is anyway less important, or contexts in which coercion threats are deemed mild, e.g. boardroom voting. We have also described a variant that retains the trackers, but having the remarkable property that now voters do not need to identify a fake tracker, and indeed the voter/tracker association can be made public (inter alia demonstrating that each voter gets a unique tracker).

We stress though that the integrity of the voters' verification checks relies on their secret trapdoor key not being compromised. We do not therefore recommend that Hyperion be used for critical, binding elections.

Full details of the constructions and proofs will appear in the full version of this paper. The authors acknowledge support of the Luxembourg National Research Fund and the Research Council of Norway for the joint project SURCVS.

References

1. Peter Y A Ryan, Peter B Rønne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In *International Conference on Financial Cryptography and Data Security*, pages 176–192. Springer, 2016.
2. Douglas Wikström. Blackbox constructions from mix-nets. Cryptology ePrint Archive, Report 2019/995, 2019. https://eprint.iacr.org/2019/995.

Modular Construction of Verified Voting Rules
Extended Abstract

Michael Kirsten

KASTEL, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
kirsten@kit.edu

Abstract. Voting rules aggregate multiple individual preferences in order to make collective decisions. Commonly, these mechanisms are expected to respect a multitude of different fairness and reliability properties, e.g., to ensure that each voter's ballot accounts for the same proportion of the elected alternatives, or that a voter cannot change the election outcome in her favor by insincerely filling out her ballot.
In this talk, I present a formal and systematic approach for the flexible and verified synthesis of voting rules from composable core modules to respect such properties by construction. Formal composition rules guarantee resulting properties from properties of the individual components, which are of generic nature to be reused for various voting rules. We provide a prototypical logic-based implementation with proofs for a selected set of structures and composition rules within the theorem prover Isabelle/HOL and a Prolog-based extension for automatic synthesis of voting rules and the corresponding proofs.

1 Introduction

In an election, voters cast ballots to express individual preferences about eligible alternatives. From these preferences, a collective decision, i.e., a set of elected alternatives, is determined using a *voting rule*. Voting rules are commonly designed to meet various expectations for fairness and reliability, but no single general rule caters for every requirement, and every rule shows paradoxical behavior for some situation [1]. The *axiomatic method* permits the analysis of desired behavior by comparing and characterizing voting rules via rigorous guarantees in the form of formal properties. Designing voting rules towards such properties is generally challenging as their trade-off is inherently difficult and error-prone.

Contribution. We present an approach for the systematic and formal design of voting rules from compact composable modules with formal properties guaranteed by construction. This work gives the core component type and compositional structures, e.g., for sequential, parallel and loop composition, and illustrates how composition rules formally establish common social choice properties. Using a Prolog-based search, we exploit the simple and rigid module structures and devise an automated pipeline that produces executable software and Isabelle proofs given the desired properties.

With the exception of the Prolog-based and automated synthesis, this and the following parts are heavily based on two previous works [5, 6].

2 Property-Oriented Composition of Voting Rules

Electoral Modules. The foundation of our approach are *electoral modules*, a generalization of voting rules. Voting rules elect a set of alternatives from a profile, i.e., a sequence of ranked ballots, and a nonempty set of alternatives A. Electoral modules are more general as they do not need to make final decisions, but instead partition A into *elected*, *rejected* and *deferred* alternatives. Hence, if an electoral module always produces a nonempty set of elected alternatives $A_{elected}$, it directly induces a voting rule which elects $A_{elected}$.

Compositional Structures. Our approach's core structures are *sequential*, *parallel* and *loop composition*, as well as the *revision* of decisions by prior modules. When composing two electoral modules

$m \triangleright n$ sequentially, the second module n only decides on alternatives which m defers and cannot reduce the alternatives already elected or rejected. A parallel composition $m||_a n$ delegates the two set-triples of m and n to an *aggregator* a, another component type which combines two such triples into one triple. Moreover, we may revise choices from prior modules and defer them for further decisions using a revision structure $\downarrow$. Finally, a loop composition $m\ \circlearrowleft_t$ reiterates a module m sequentially until either m's iteration reaches a fixed point, or a *termination condition* t holds, i.e., a component type which is simply a predicate on a triple of sets of alternatives.

A Simple Example. The well-known *Baldwin's rule* [2] can be sequentially composed with a loop structure of a module eliminating the alternative with the lowest Borda score and terminating when only one alternative remains, and a module which elects all deferred alternatives. This construction directly establishes, e.g., the Pareto property and Condorcet consistency as the loop may never reject a Condorcet winner and always rejects Pareto-dominated alternatives.

3 Automated Synthesis for Trusted and Executable Voting Rules

A Component Library. Given our simple and rigid module structure, we devised components and composition rules for various well-known voting rules, including representative examples for Condorcet and scoring rules, as well as filtering modules that are useful for the prominent class of knockout tournaments. We implemented this library within Isabelle/HOL together with selected composition rules that enable proofs for social choice properties such as the Condorcet criterion, monotonicity, homogeneity, reinforcement, and anonymity.

Translation to Prolog and Automated Search. Consequently, we take the type information, assumptions and proved lemmas from functions and theorems of the Isabelle code and translate it to simple Prolog formulas. Using the built-in Prolog search, this yields a decidable program that, given the desired social choice properties, produces voting rule compositions for which the properties are fulfilled.

Automated Synthesis of Voting Rules. We can, moreover, play this back to our Isabelle code in order to verify the obtained composition directly within Isabelle. Besides a readable Isabelle proof, given that the modules and structures themselves are comparatively simple, we can also use Isabelle to generate verified and executable programs [9], e.g., in the form of Scala code, for the found composition. With a few more optimizations to the Prolog search algorithm, we end up with a flexible pipeline that, given the desired properties in higher-order logic, uses the rules, components, and structures in the Isabelle framework, builds a synthesis tool with executable code and a trustworthy proof that the synthesis is correct.

4 Related Work and Conclusion

Related Work. Our electoral modules are based on less-formal components for hierarchical electoral systems from [4]. Other work designs voting rules less modularly for statistically guaranteeing social choice properties by machine learning [10]. Prior modular approaches target verification [7] or declarative combinations of voting rules [3], but ignore social choice properties. Specific compositional structures as presented in [8] are readily expressible by our structures.

Conclusion. Our approach enables flexible and intuitive compositions of voting rules from a small number of structures with precise and general interfaces, easily extended with further modules. This allows to formally establish common social choice properties from given component properties by rigorous composition rules. With our Prolog-based extension, we obtain a powerful automated synthesis tool that gives us verified program code that implements a voting rule which fulfills the queried social choice properties.

References

1. Arrow, K.J.: Social Choice and Individual Values. No. 12 in Monographs / Cowles Commission for Research in Economics, Wiley, New York, USA (1951)
2. Baldwin, J.M.: The technique of the Nanson preferential majority system of election. Transactions and Proceedings of the Royal Society of Victoria **39**, 42–52 (1926)
3. Charwat, G., Pfandler, A.: Democratix: A declarative approach to winner determination. In: Walsh, T. (ed.) 4th International Conference on Algorithmic Decision Theory (ADT 2015). Lecture Notes in Computer Science, vol. 9346, pp. 253–269. Springer (2015). https://doi.org/10.1007/978-3-319-23114-3_16
4. Grilli di Cortona, P., Manzi, C., Pennisi, A., Ricca, F., Simeone, B.: Evaluation and Optimization of Electoral Systems. SIAM (1999)
5. Diekhoff, K., Kirsten, M., Krämer, J.: Formal property-oriented design of voting rules using composable modules. In: Pekeč, S., Venable, K. (eds.) 6th International Conference on Algorithmic Decision Theory (ADT 2019). Lecture Notes in Artificial Intelligence, vol. 11834, pp. 164–166. Springer (2019). https://doi.org/10.1007/978-3-030-31489-7
6. Diekhoff, K., Kirsten, M., Krämer, J.: Verified construction of fair voting rules. In: Gabbrielli, M. (ed.) 29th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2019), Revised Selected Papers. Lecture Notes in Computer Science, vol. 12042, pp. 90–104. Springer (2020). https://doi.org/10.1007/978-3-030-45260-5_6
7. Ghale, M.K., Goré, R., Pattinson, D., Tiwari, M.: Modular formalisation and verification of STV algorithms. In: Krimmer, R., Volkamer, M., Cortier, V., Goré, R., Hapsara, M., Serdült, U., Duenas-Cid, D. (eds.) Third International Joint Conference on Electronic Voting (E-Vote-ID 2018). Lecture Notes in Computer Science, vol. 11143, pp. 51–66. Springer (2018). https://doi.org/10.1007/978-3-030-00419-4_4
8. Narodytska, N., Walsh, T., Xia, L.: Combining voting rules together. In: Raedt, L.D., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) 20th European Conference on Artificial Intelligence (ECAI 2012). Frontiers in Artificial Intelligence and Applications (FAIA), vol. 242, pp. 612–617. IOS Press (2012). https://doi.org/10.3233/978-1-61499-098-7-612
9. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic, Lecture Notes in Computer Science, vol. 2283. Springer (2002). https://doi.org/10.1007/3-540-45949-9
10. Xia, L.: Designing social choice mechanisms using machine learning. In: Gini, M.L., Shehory, O., Ito, T., Jonker, C.M. (eds.) International conference on Autonomous Agents and Multi-Agent Systems (AAMAS '13). pp. 471–474. IFAAMAS (2013), http://dl.acm.org/citation.cfm?id=2484995