Karlsruhe Institute of Technology

Institute of Theoretical Informatics (ITI)
Application-oriented Formal Verification

Automated Verification for Functional and Relational Properties of Voting Rules [COMSOC 2016]

Bernhard Beckert, Thorsten Bormer, Michael Kirsten, Till Neuber, and Mattias Ulbrich

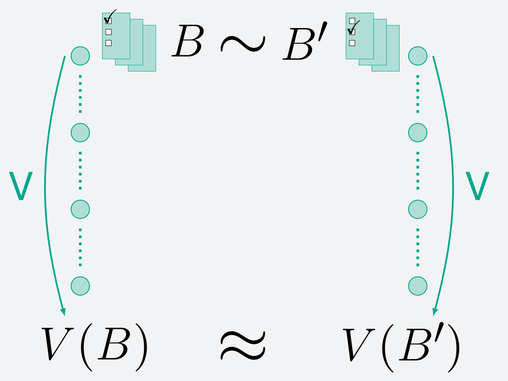
Motivation

- Voting rules often required (e.g., by constitution) to fulfil axiomatic properties
- Design of voting rules with desired properties **non-trivial** and **error-prone**
- **Growing complexity** with rise of electronic voting increases vulnerability
- Solution: Computer-aided verification for **trustworthy** voting rules

Relational Specification: Coupling Evaluations

- Technique for proving relational (**inter-profile**) properties, e.g., anonymity
- Relational properties consider two ballot profiles and election outcomes

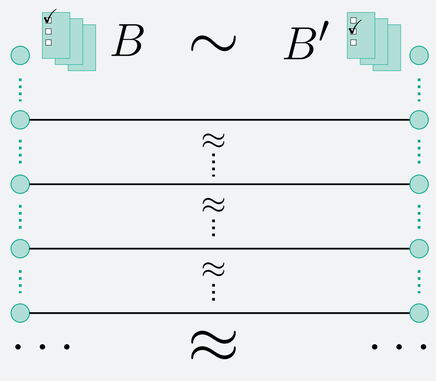
Separate Evaluations



Example

$\max_c \Sigma_{i=0}^{N} B_{i,c} = \max_c \Sigma_{i=0}^{N} B'_{i,c}$

Coupling Evaluations



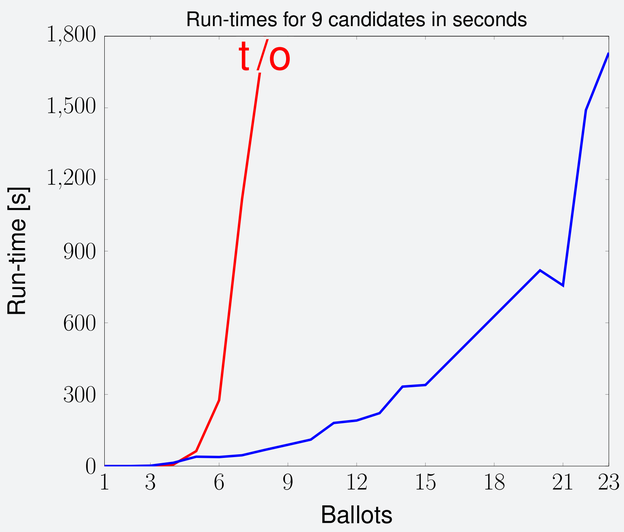
Example

`result1 = result2`

- Often enables short and concise specifications (only **differences**)
- Critical point for making **verification feasible!**

Relational Verification: Examples

Verification using Bounded Model Checking (Tool: CBMC)



- Separate / coupling eval.
- Achieved higher bounds!
- Anonymity Prop.: Indifference to **renaming and permutation of voters**
- Plurality Rule: Single choice, candidate with plurality of votes is elected
- Concise specifications useable for BMC $\Rightarrow$ Guidance for SAT-solver

Verification using Deductive Theorem Proving (Tool: KeY)

	Plurality Rule	Approval Rule	Range Rule	Borda Count
Anonymity	33	43	44	44
Neutrality	42	56	57	57
Monotonicity	46	47	48	52
Participation	28	50	51	50
Homogeneity	53	70	71	71

- Verified various properties (numbers are required lines of specification)
- Proof construction almost fully automatic (< 10 user interactions)
- Verification using separate evaluations often not feasible

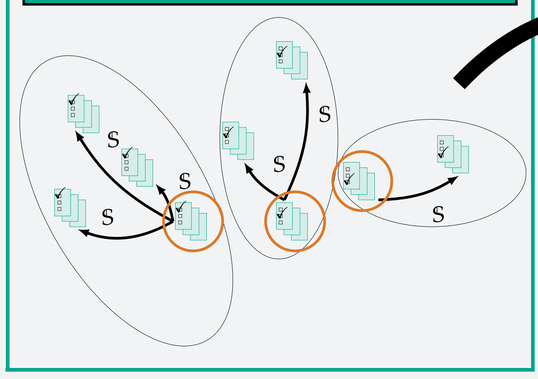
Verifying Voting Rules

- Formalisation: Rules as imperative algorithms (C / Java), properties in $\text{FOL}_{\mathbb{N}}$
- **Established** verification techniques: **KeY** and **CBMC**
  $\Rightarrow$ Deductive Theorem Proving and Bounded Model Checking (BMC)

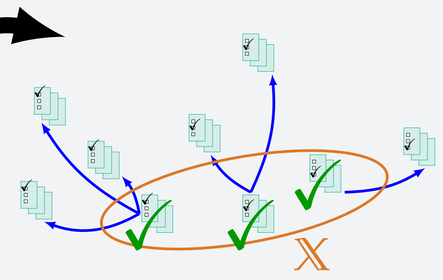
Functional Specification: Exploiting Symmetries

- Already established symmetry, target: functional (**intra-profile**) property
- Functional property considers elections individually, e.g., majority criterion
- Symmetry example: Anonymity, operation is ballot permutation

Symmetric profiles have minimal elements



Verify only elements in $\mathbb{X}$

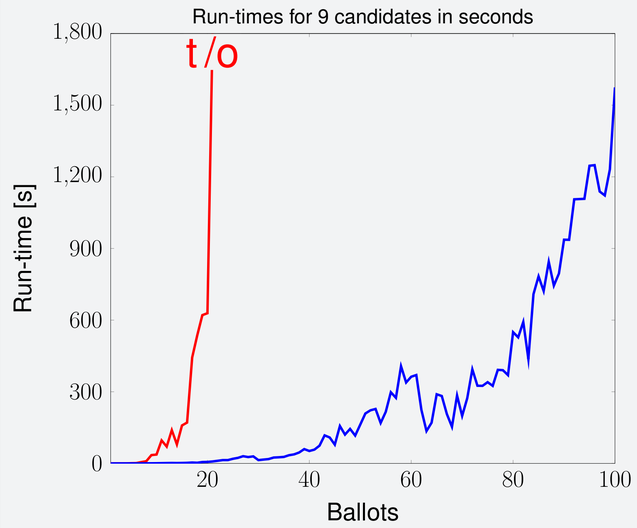


- Minimal elements form a set $\mathbb{X}$
- All possible profiles reachable from $\mathbb{X}$

- Symmetry properties infer **symmetry-breaking predicates** (SBPs)
- Reduces search space (to $\mathbb{X}$) using SBP as precondition
- Example for anonymity: Check **only sorted** (by candidate) **profiles**

Functional Verification: Example

Verification using Bounded Model Checking (Tool: CBMC)



- With / without SBP (anon.)
- Pushed the boundaries!
- Majority Criterion: If candidate c has majority, c must be elected
- Plurality Rule: Single choice, candidate with plurality of votes is elected

General Approach for Functional Verification

- Verification Task: Does voting rule V satisfy property P ?
- Conjecture: V satisfies symmetry property $\mathbb{S}$.

General Approach

1. Verify $\mathbb{S}$ for V using relational techniques
2. Verify V satisfies property P only for subset $\mathbb{X}$
3. Prove that $\mathbb{X}$ spans **all possible profiles** (independent of V!)
4. Prove that $\mathbb{S}$-operations preserve property P (independent of V!)

Conclusion

- **General approach**: Verification of functional axiomatic properties
- **Feasibility demonstrated** on multiple well-known results