UMassAmherst

Specifying and Verifying Requirements for Election Processes

Borislava I. Simidchieva, Matthew S. Marzilli, Lori A. Clarke, Leon J. Osterweil

Laboratory for Advanced Software Engineering Research (LASER)
Computer Science Department
University of Massachusetts, Amherst
140 Governors Drive
Amherst, MA 01003

Motivation

- Elections entail far more than the casting and tabulation of votes
- Need to consider the entire process, in which voting machines play only only part, out of many
- The election process is large and complex and, in the U.S., varies from jurisdiction to another

Goal

- To detect potential defects in election processes and identify ways to correct them

High-level approach

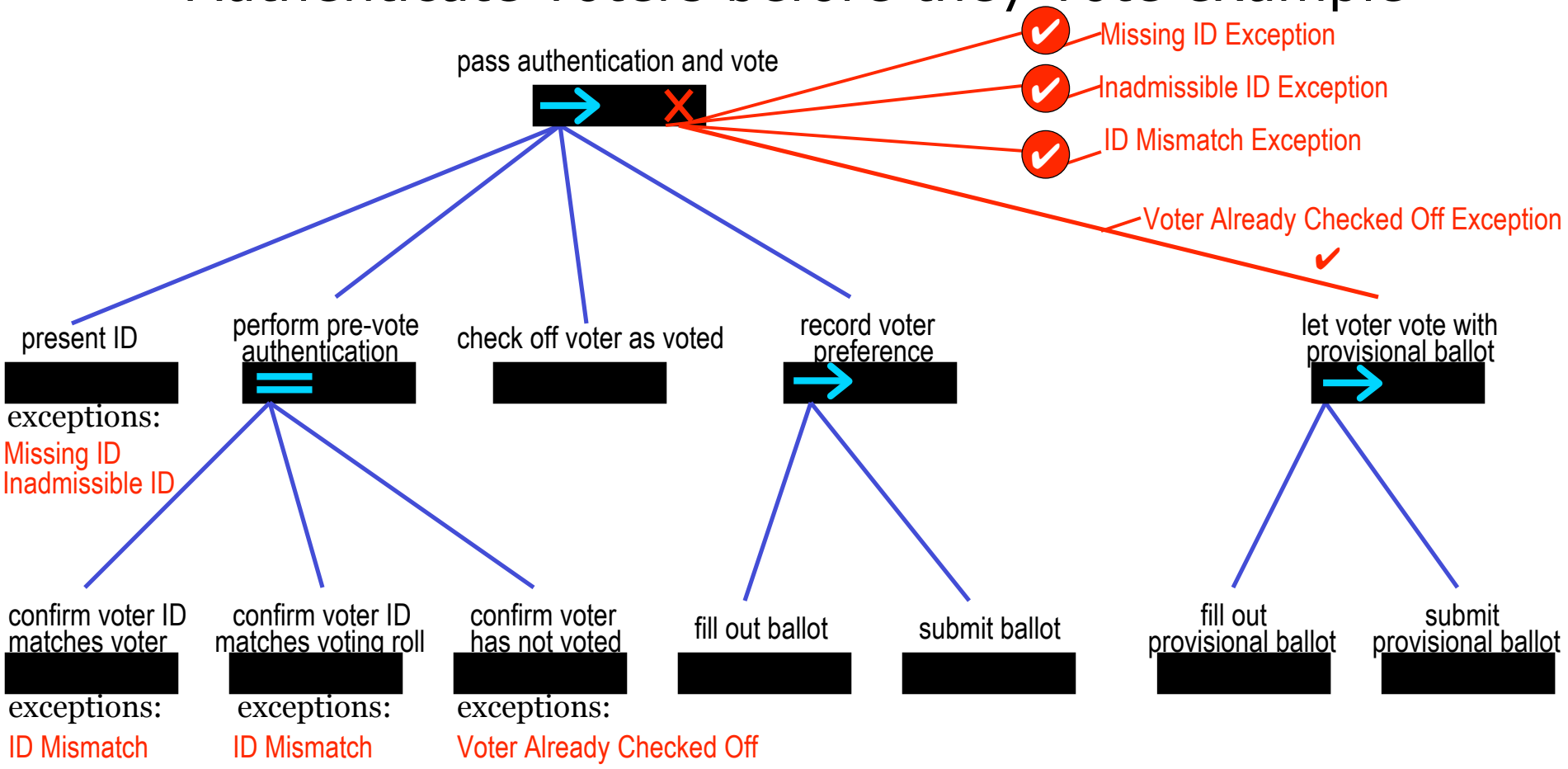
- Define an example election process
  - Coordination of activities, or steps
  - Agents, resources, and more
- Refine the requirements for an election process
  - High-level requirements
  - Low-level requirements
  - Precise properties, or event sequences
- Verify the process adheres to the properties
  - Define correspondence between property events and process steps
  - Run formal analysis using finite-state verification

Define the election process

- Use the Little-JIL process definition language
  - Consists of coordination diagram, and other specifications (e.g. agents, artifacts, resources)
  - Especially appropriate for modeling concurrency and complex exception handling that arise in elections
  - Visual representation facilitates communication and validation
- Three main example election process phases:
  - Pre-polling activities that happen before election day
  - Activities that occur on election day
  - Counting of votes after election completes

Example election process

- Authenticate voters before they vote example



Refine the requirements

- High-level requirements
  - Technology-independent
  - Usually applicable to most election processes
  - E.g. “Each unique voter is allowed at most one vote”
- Low-level Requirements
  - Refine each high-level requirement into a collection of low-level requirements
- Formal Properties
  - Each low-level requirement must be formally defined as a property
  - A property must be in a form suitable for verification, such as a finite-state automaton (FSA)
  - Use PROPEL to guide the property definition and generate the FSA
  - PROPEL provides three equivalent, synchronized views: question tree, disciplined English, and finite-state automaton

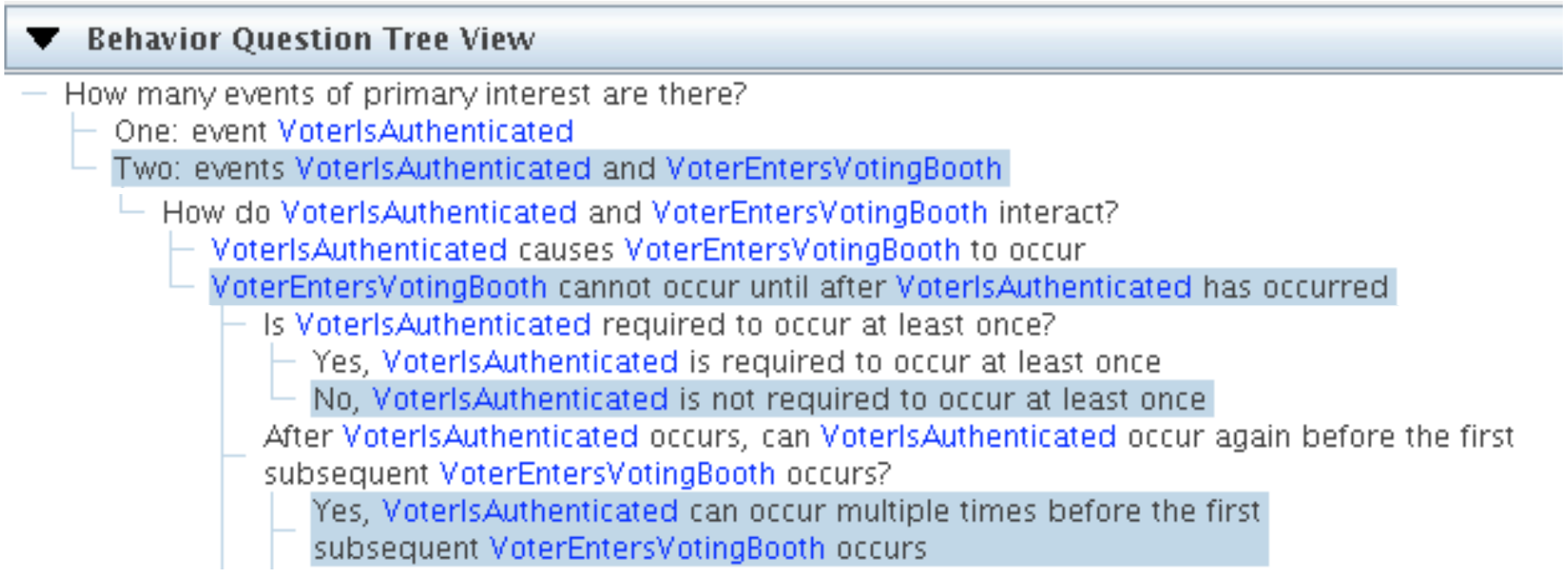
Decompose high-level requirements

- Example refinement of high-level requirement into a collection of low-level requirements

each unique voter is allowed at most one vote
➔ voter must be authenticated before entering voting booth
➔ voter must be checked off before entering voting booth
➔ voter must enter voting booth before choosing to vote
➔ voter must receive ballot before choosing to vote
➔ voter must leave voting booth after choosing to vote

Formally define the properties

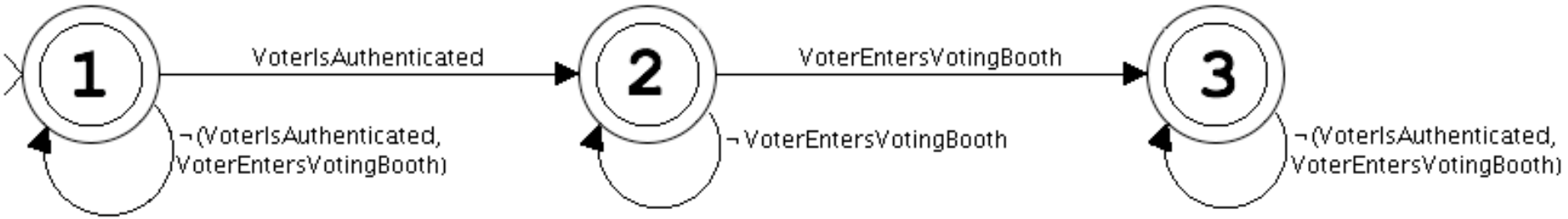
Use the PROPEL property elicitation tool to formally define a property corresponding to the low-level requirement "voter must be authenticated before entering voting booth"



Example property

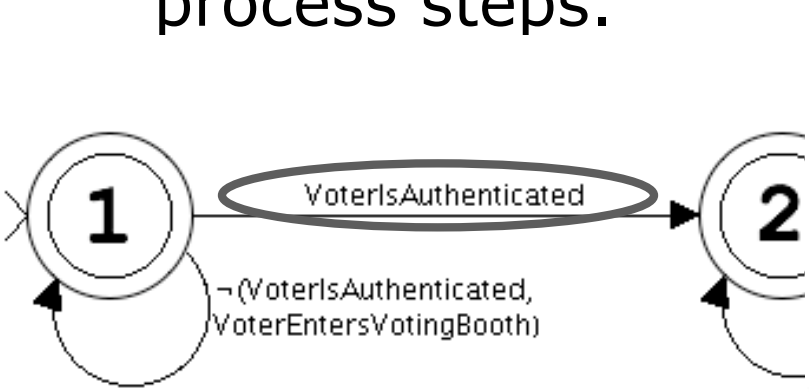
Voter must be authenticated before entering voting booth:

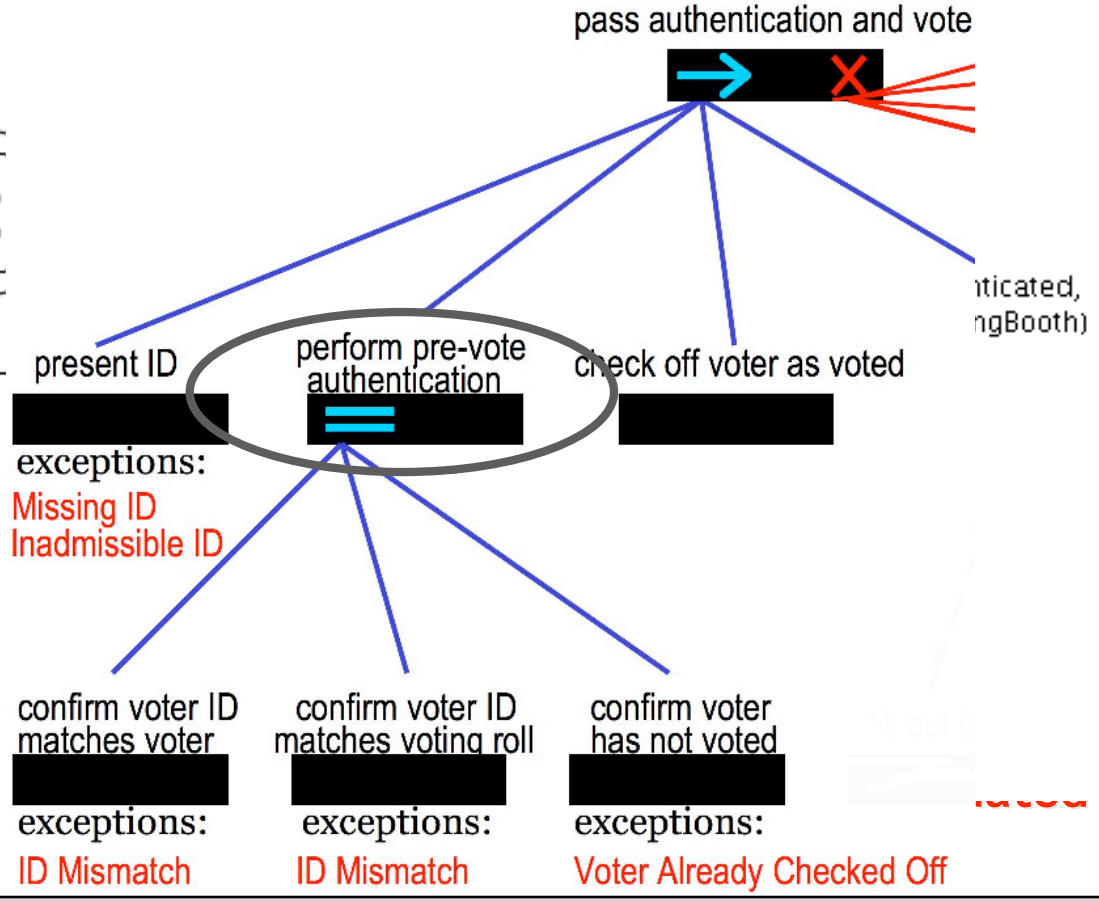
- Disciplined English view:
  - ***VoterEntersVotingBooth*** *cannot occur until after* ***VoterIsAuthenticated*** *has occurred.* ***VoterIsAuthenticated*** *is not required to occur, however.*
  - ***VoterIsAuthenticated*** *can occur multiple times before the first subsequent* ***VoterEntersVotingBooth*** *occurs.*
  - *After* ***VoterIsAuthenticated*** *occurs other events can occur before the first subsequent* ***VoterEntersVotingBooth*** *occurs*
  - *After* ***VoterEntersVotingBooth*** *occurs neither* ***VoterIsAuthenticated*** *nor* ***VoterEntersVotingBooth*** *can occur again.*
- FSA view:



Binding property events to process steps

For verification, property events must be bound to process steps.





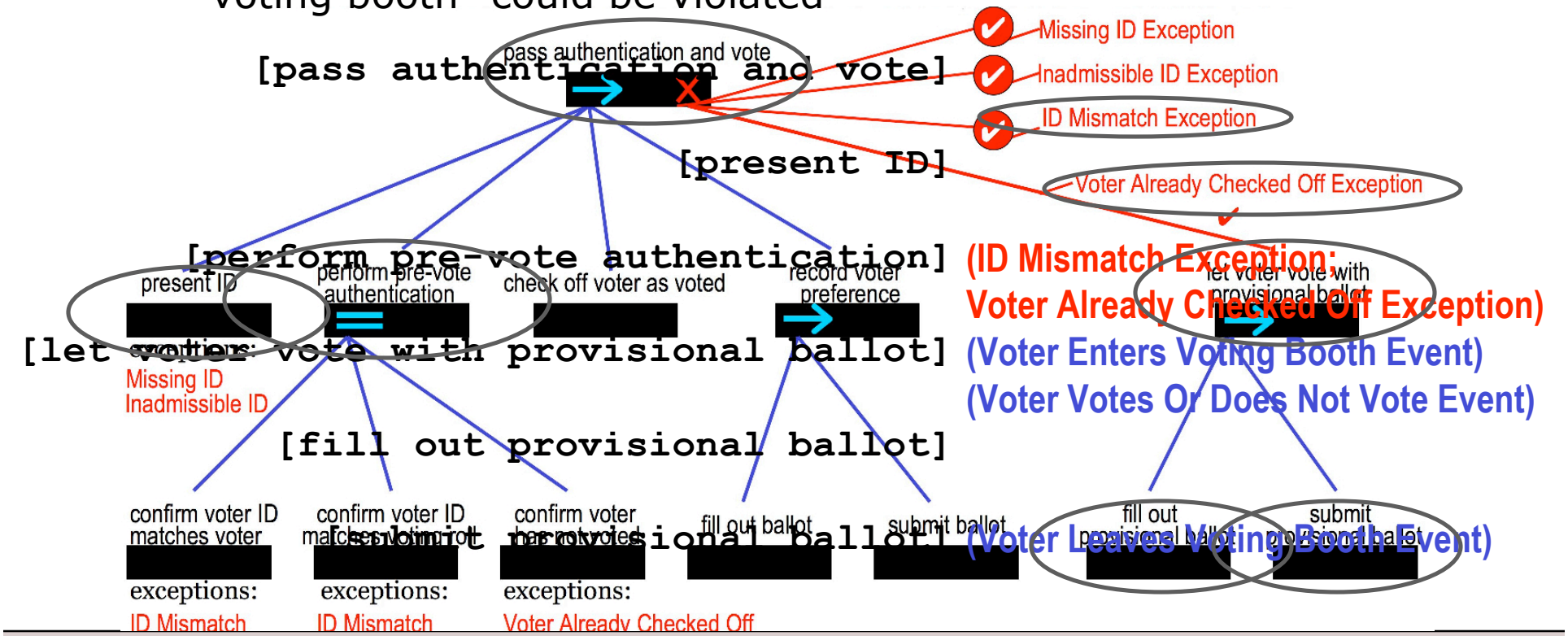
satisfies the property

Finite-state verification (FSV) with FLAVERS

- The FLAVERS FSV verifier has been extended to automatically construct finite models of the Little-JIL process definitions
- Finite model represents all possible event sequences, for the events in a property, that could occur for all the possible traces through the process definition
- Apply dataflow analysis algorithm to determine if the model is consistent with the property
- If the process is inconsistent with the property, a counter-example trace is produced
- FLAVERS determines whether the election process, as defined in Little-JIL, adheres to the property "voter must be authenticated before entering voting booth"

Violation detected

- An unauthenticated voter can vote with provisional ballot
  - Counter-example produced by FLAVERS to demonstrate how the property "voter must be authenticated before entering voting booth" could be violated



Violation detected, cont'd

- Property violation possible because pre-vote authentication step is executed in parallel and exceptions can occur in any order
- Forcing sequential execution can correct this error
- After correcting the process definition, the FLAVERS verifier is run again to verify that the new process definition satisfies the "voter must be authenticated before entering voting booth" property, as well as the other properties

Observations

- Formal process definition to rigorously and precisely capture the election process provides:
  - A *holistic* view of the election process
  - Extendability by further elaborating the process definition in Little-JIL
- Precise requirement specification plays a pivotal role in ensuring the correctness and fairness of elections
  - Refining high-level requirements into low-level properties is difficult
  - The PROPEL property elicitation tool facilitates the refinement
- Formal verification can determine with certainty whether an election process definition adheres to specific properties
  - FLAVERS produces a counter-example trace if a property is violated
  - The counter-example trace can help to improve the election process definition

Related Work

- Direct Recording Electronic (DRE) machines
  - ACCURATE; Brennan Center for Justice; EVEREST; Caltech/MIT Voting Technology Project
- Verification of elections
  - Mercuri & Neumann; Saltman
- Requirements for elections
  - Mitrou; Lambrinoudakis et at
- Election processes
  - Election Assessment Hearing; Raunak

Future Work

- Specify process desiderata
  - Technology-independent, high-level requirements that most election processes should meet
  - Corresponding properties may differ for different processes
- Organize properties and reason about their relationships
  - E.g. approving a provisional ballot versus protecting voter confidentiality
- Fault-tree analysis
  - Considers how the incorrect processing of a step can cause an error
  - Useful for identifying single points of failure (i.e. a single step in the election process definition that, if executed incorrectly, may lead to the election being compromised)

Questions?

Election processes are too important to democracy to not be carefully defined and evaluated