Open Problem: Complex Voting Schemes

A Formally Verified Single Transferable Vote Scheme with Fractional Values

Milad Ghale, Rajeev Goré, Dirk Pattinson
Australian National University

Rajeev.Gore@anu.edu.au

October 26, 2017
Overview

E2E Verifiabiity Needs Program Verification
Single Transferable Voting (STV) scheme?
Why is it hard to tally ballots according to STV?
Current computer counting in Australia
Where is the scrutiny and trust?
Interactive Synthesis of Vote Counting Programs
Results, Features, Further Work, Caveats and Conclusion
E2E Verifiability Needs Program Verification

Cast as intended: voters verify that electronic ballot is correct
Recorded as cast: ballot was not tampered with in transit
Tallied as recorded: voter can verify that ballot was tallied
Cast as intended: voters verify that electronic ballot is correct
Recorded as cast: ballot was not tampered with in transit
Tallied as recorded: voter can verify that ballot was tallied
But … what if the vote-counting program contains bugs?
Software independence:
Cast as intended: voters verify that electronic ballot is correct
Recorded as cast: ballot was not tampered with in transit
Tallied as recorded: voter can verify that ballot was tallied
But ... what if the vote-counting program contains bugs?
Software independence:

Idea 1: vote-counting programs must produce a tallying script
Idea 2: if the tallying script is correct then the result is correct
Idea 3: it is trivial to write a program to check tallying script
What do we mean by voting scheme?

A method for setting out, filling in and counting ballots

### STV Ballot Form

Rank any number of candidates in order of preference.

<table>
<thead>
<tr>
<th>Candidate</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dave</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Setting out:** order of candidates fixed or Robson rotated?

**Filling in:** write all numbers from 1 to \( N \) or only ones you want?

**Counting:** quota required to be elected; who is weakest candidate; how to break ties; how to transfer a vote; when to stop counting

Nothing to do with electronic voting . . . yet

In particular, nothing to do with security aspects of e-voting
Single Transferable Vote Counting is Non-trivial

**Vacancies**: number of candidates that we need to elect
**Candidates**: number of people standing for election
**Quota**: how many votes are required to elect a candidate
**Ballot**: is a vote for highest ranked continuing candidate
**Counting**: proceeds in rounds
**Surplus**: ballots are transferred to next continuing candidate
**Transfer Value**: current value of ballot (possibly $\leq 1$
**Eliminate Weakest**: but how to break ties

### STV Ballot Form

**Rank any number of candidates in order of preference.**

Alice 3
Bob  
Charlie 1
Dave 2

**Rounds**: repeat until all seats filled
**Tally**: all highest preferences
**Elected**: All candidates with “quota” are elected
**Eliminated**: If nobody elected this round then eliminate weakest candidate
**Transfer**: compute new transfer values
**Autofill**: If can seat all remaining candidates, do so
Example

Droop Quota: \( Q = \left\lfloor \frac{\text{totalnumberofballots}}{\text{seats}+1} \right\rfloor + 1 \)

Candidates: \( A, \ B, \ C, \ D \)
Seats: 2
Ballots: 5

\[ A > B > D \]
\[ A > B > D \]
\[ A > B > D \]
\[ D > C \]
\[ C > D \]

Assume no fractional transfers and no autofill
Example

Droop Quota: \( Q = \left\lfloor \frac{\text{total number of ballots}}{\text{seats}+1} \right\rfloor + 1 \)

Candidates: \( A, B, C, D \)

Seats: 2

Ballots: 5

\[ Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2 \]

\( A > B > D \)
\( A > B > D \)
\( A > B > D \)
\( D > C \)
\( C > D \)

Elected: \( A, D \)

Eliminated: \( B \)

Assume no fractional transfers and no autofill
Example

Droop Quota: \[ Q = \left\lfloor \frac{\text{totalnumberofballots}}{\text{seats}} + 1 \right\rfloor + 1 \]

Candidates: \( A, \ B, \ C, \ D \)

Seats: 2

Ballots: 5

\[ Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2 \]

\[ \begin{align*}
A & > B > D \quad \text{votes}(A) = 1 \\
A & > B > D \quad \text{votes}(A) = 2 \\
A & > B > D \quad \text{votes}(A) = 3 \\
D & > C \quad \text{votes}(D) = 1 \\
C & > D \quad \text{votes}(C) = 1
\end{align*} \]
Example

Droop Quota: \( Q = \left\lfloor \frac{\text{totalnumberofballots}}{\text{seats}+1} \right\rfloor + 1 \)

Candidates: \( A, B, C, D \)

Seats: 2

Ballots: 5

\[
Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2
\]

\[
A > B > D \\
A > B > D \\
A > B > D \\
D > C \quad \text{votes}(D) = 1 \\
C > D \quad \text{votes}(C) = 1
\]

Elected: A
Example

Droop Quota: \[ Q = \left\lfloor \frac{\text{totalnumberofballots}}{\text{seats} + 1} \right\rfloor + 1 \]

Candidates: \( A, B, C, D \)
Seats: 2
Ballots: 5

\[
A > B > D \\
A > B > D
\]

\( D > C \quad \text{votes}(D) = 1 \)
\( C > D \quad \text{votes}(C) = 1 \)

Elected: \( A \)
Example

Droop Quota: \( Q = \left\lfloor \frac{\text{total number of ballots}}{\text{seats} + 1} \right\rfloor + 1 \)

Candidates: \( A, \ B, \ C, \ D \)

Seats: 2

Ballots: 5

\[ Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2 \]

\( A > B > D \)
\( A > B > D \)
\( X > B > D \) \( \text{votes}(B) = 1 \)
\( D > C \) \( \text{votes}(D) = 1 \)
\( C > D \) \( \text{votes}(C) = 1 \)

Elected: \( A \)
Example Droop Quota: 

\[ Q = \left\lfloor \frac{\text{total number of ballots}}{\text{seats}+1} \right\rfloor + 1 \]

Candidates: \( A, B, C, D \)

Seats: 2
Ballots: 5

\[
Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2
\]

\[ A \succ B \succ D \]
\[ A \succ B \succ D \]
\[ X \succ X \succ D \quad \text{votes}(D) = 2 \]
\[ D \succ C \]
\[ C \succ D \quad \text{votes}(C) = 1 \]

Elected: \( A \)
Eliminated: \( B \)
Example

Droop Quota: \[ Q = \left\lfloor \frac{\text{totalnumberofballots}}{\text{seats}+1} \right\rfloor + 1 \]

Candidates: \( A, B, C, D \)

Seats: 2

Ballots: 5

\[ Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2 \]

\[ A \succ B \succ D \]

\[ A \succ B \succ D \]

\[ A \succ B \succ D \]

\[ A \succ B \succ D \]

\[ A \succ B \succ D \]

\[ A \succ B \succ D \]

\[ A \succ B \succ D \]

votes(D) = 2

\[ D \succ C \]

\[ C \succ D \]

votes(C) = 1

Elected: \( A, D \)

Eliminated: \( B \)
Existing Electronic Vote-counting in Australia

Australian Electoral Commission: proprietary code; not available for scrutiny; FOI request to publish code denied on grounds of “security” and “commercial in confidence”

Victorian Electoral Commission: proprietary code; available for scrutiny; no formal scrutiny to my knowledge

Australian Capital Territory: eVACS™
  - developed by Software Improvements Pty Ltd. using C++
  - used since 2001 to count four elections
  - counting code available from ACTEC website
  - full code available if you sign a confidentiality agreement

New South Wales Electoral Commission: detailed functional requirements publicly available; found to comply with legislation by legal expert from QUT; certified by Birlasoft as passing all tests; proprietary code; code not available for scrutiny

TM eVACS is a trademark of Software Improvements Pty Ltd.
ACTEC and SoftImp Approach

- **scrutiny**
- **artifacts**
- **trust**

- **published**
- **legal text**

- **ACTEC**
- **SoftImp**

- **published?**
- **functional specs using UML**

- **evidence?**
- **SoftImp**

- **semi-published**
- **computer code**

- **ACTEC & SoftImp**
  - “audited” by BMM: all okay
NSWEC Approach

- **NSWEC**
  - Published legal text
  - Published 47 pages of functional specs with flow chart

- **Vendor**
  - Evidence?
  - Published proprietary computer code

- **legal expert (QUT)**
  - Audited by Birlasoft: passed all tests

- **trust**
Bugs in ACT and NSW Counting Modules

ANU logic group: found three bugs in eVACS
  programming error: simple for-loop bounds error
  ambiguous legal text: break weakest candidate ties by inspecting previous round where “all candidates have an unequal number of votes”
  programming error: un-initialised boolean: different compilers give different results
  how bad: for every bug, we could generate an election in which the code gave the wrong result

UniMelb group: found bug in NSWEC code whereby one candidate’s chances of winning were reduced from 90% to 10% and she lost the 2015 election! No recourse as the three month period for a legal challenge had passed.
“Simplifications” in ACT Legislation Are Harmful

ANU logic group: we showed last year that
Rounding (fractions): errors can become significant
Point of declaring winners: can be significant
“Last parcel” simplification: is just silly
How bad: for every “simplification”, there is an election where legislation gave the wrong result w.r.t. Vanilla STV
Efficient Interactive Synthesis Via Mathematical Proof

- scrutiny
- published

- artifacts
  - legal text
  - manual

- trust
  - (your) elections expert

- rules capturing
  - state transitions
  - of counting process
  - manual

- functional specs as
  - formula of typed
  - higher-order intuitionistic logic
  - manual

- automatic!

- published

- certificate producing
  - computer code
  - published

- published Coq certificate
  - Coq proof:
    - correct certificate implies correct count
Features

Completed: STV vote-counting and Schulze Method

Exact fractions: our code for STV manipulates fractions exactly

Efficiency: can (STV) count up to 10 million votes with 40 candidates and 20 vacancies in 20 minutes

Certificate: our code produces a (plain text) certificate that vouches for the correctness of the count

Scrutiny: average third year CS student can write a program to check the certificate is correct w.r.t. published rules and published ballots

Trust: you don’t even need to trust the hardware or software since a correct certificate implies a correct count
Why Should We Trust Machine-checked Proof?

1930s Alonzo Church’s typed λ-calculus

INRIA

→

Coq theorem prover: 50K lines of OCaml code

Coq development team

OCaml compiler and hardware

peer review

logic community

trust

scrutiny

artifacts

published

published

published

person

proof checked by Coq
Further Work, Caveats and Conclusions:

**Verified Certificate Checker:** using CakeML to verify our certificate checker against a formal model of the semantics of C

**Other flavours of STV:** cover all STV schemes used in Australia

**Effort:** approximately 4 person-months of work by a Coq novice

**Caveat:** relies on EMB publishing the ballots in clear text so it is vulnerable to the Sicilian Attack

**Shufflesum:** currently trying to synthesise the code

**Conclusion:** verified synthesis possible for complex e-counting
Jobs in Computer Science at ANU!

Fellowships: 5 years fixed term for early career researchers

Tenure possibilities: high but depend upon your performance and our budget

Deadline: 5th of November


Contact: me if you want to chat about them