Democracy as a Critical System:
Formal Methods for Voting

Joseph Kiniry
IT University of Copenhagen
Research Style

- novel, difficult topics
- open source tools usable by undergraduates
- basic research
- mathematics biased toward usable tools
- applied research
Current Research Projects

- self-healing systems
- seamless architectures
- verified elections

- formal methods for sensor networks
- specification and verification of concurrent Java
Formal Methods
“Pure”
Formal Methods
“Applied”
Formal Methods
Critical Systems
Critical Systems
Democracy
Activism and Science
Computer Scientist

Mathematician
Voting Theory
Computers in Elections
Electronic Voting
e-Voting
Voting Machines

- punchcard ballots
- physical locks
- lever machines
- dedicated primitive hardware
- mechanical ballot boxes
- off-the-shelf Windows machines
- Tuesday, 8 March 2011
e-Voting Worldwide
e-Voting Worldwide

Tuesday, 8 March 2011
Computer-based Voting in The Netherlands

dedicated computer-based voting machines since the late 90s
people generally trust the government
experiments in remote voting for expats
hacking an election

recommendations to the government
tally system developed with formal methods
KOA

Tuesday, 8 March 2011
Computer-based Voting in Ireland

PR-STV novel social vote counting

last-minute secret purchase of €40M in Nedap machines

PowerVote independent system testing

CEV

Vótáil scrapping e-voting at a cost of €55M

Tuesday, 8 March 2011
Computer-based Voting in Denmark

People generally trust the government.

Claim: No computers are used in voting.

In truth: closed-source tally system used to compute final outcome.

Regular proposals to introduce e-voting.

DVT

E-voting trials at the local level.

DemoTech
most open source voting systems are not tested
most proprietary voting systems are not tested

Testing Voting Systems

“hard-core” testing is random testing is no testing
random testing of multiple implementations

how does one rigorously test a voting system?

Tuesday, 8 March 2011
Relating The Law to Software
The State of e-Voting Software Today
1.2.2. Step Two: Determining of Passing the Threshold

This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of three requirements. Thus, the Danish electoral system has not one, but three different electoral thresholds, and parties qualify for participation in the allocation of compensatory seats by any one of them. The three thresholds are:

1. winning a seat directly in any of the ten multi-member constituencies,
2. obtaining in two of the three electoral provinces a number of votes corresponding, at least, to the provincial vote-seat ratio fixing in the calculation of these ratios the number of seats in the multi-member constituencies in the electoral provinces in question, excluding the provinces’ compensatory seats; or
3. 2 per cent of the valid, national vote.

For parties that do not meet the first requirement (in 2007 it was two of nine participating parties), the relevant numbers are shown in Table 2, which allows a comparison of thresholds (2) and (3), and the votes for the two parties in question in the three electoral provinces as well as nationally.

Experience shows that threshold (3), the 2 per cent rule, is much more important than threshold (2), the vote-seat ratio in two of three electoral provinces. Parties that meet the 2 per cent requirement will often also have met threshold (2) – as was the case in 2007 with the Unity List – while parties below the 2 per cent hurdle almost invariably will not meet any of the other requirements (as shown by the example of the Christian People's Party in 2007, which failed to cross any of the three thresholds). This experience illustrates how Danish political parties are not (any longer) primarily local or provincial in their support patterns.

1.2.3. Step Three: Allocating Compensatory Seats to Parties

This is the decisive step, since it is here that the proportional, overall, national (or upper tier) allocation of all 175 seats takes place. The calculation (reproduced in Table 3 below) allocates the seats available to parties which have qualified for participation in this allocation in strict proportionality to the number of votes obtained by these parties. The calculation is done on the basis of the so-called pure rare quota; seats not allocated by the full
char*M,A,Z,E=40,J[40],T[40];main(C){for(*J=A=scanf(M="%d",&C);--
            E;             J[E]=T
[40]
=E)  printf("._");  for(;(A-=Z=!Z)  ||  (printf("\n\n")
    ,   A    =              39              ,C             --
)    ;   Z    ||    printf   (M   ))M[Z]=Z[A-(E   =A[J-Z])&&!C
&    A   ==             T[                                  A]
Reflection Relation
In our tests, it counts correctly.

Overall Correctness

Argument

Trust us, it works. How hard can it be, adding one over and over?
The State of Verified e-Voting Software Today
The Law

1.2.2. Step Two: Determining of Passing the Threshold

This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of the requirements. The proportional system has not one, but three different electoral thresholds, and parties qualify for participation in the allocation of compensatory seats by any one of them. The three thresholds are:

1. winning a seat directly in any of the ten multi-member constituencies;

2. obtaining in two of the three electoral provinces a number of votes corresponding to at least 2% of the national votes/seat ratio in the multi-member constituencies in the electoral provinces in question, excluding the provinces' compensatory seats; or

3. 2% of the valid, national vote.

Experience shows that threshold (3), the 2% per cent rule, is much more important than threshold (2), the vote/national seat ratio in the electoral provinces. Parties that meet the 2% per cent requirement will often also have met threshold (2) - as was the case in 2007 with most seats in seat while parties below the 2% hurdle almost invariably will not meet any of the other requirements as shown by the example of the Christian People's Party in 2007, which failed to cross any of the three thresholds. This experience illustrates how political parties are not (any longer) primarily local or provincial in their support patterns.

1.2.3. Step Three: Allocating Compensatory Seats to Parties

This is the decisive step, since it is here that the proportional, overall, national (or upper-tier) allocation of all 175 seats takes place. The calculation (reproduced in Table 3 below) allocates the seats available to parties who participate in this allocation. The result is a strict proportional representation of the number of votes obtained by these parties. The calculation is done on the basis of the so-called pure quota; seats not allocated by the full

Table 2. How the Parties that Failed to Qualify for Seats at Threshold (1) Fared on Threshold (2) and (3), November 13, 2007.

<table>
<thead>
<tr>
<th></th>
<th>All of Denmark</th>
<th>Metropolitan Copenhagen</th>
<th>Sealand-Southern Jutland</th>
<th>Northern and Central Jutland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold 2:</td>
<td></td>
<td>26,908</td>
<td>25,103</td>
<td>25,146</td>
</tr>
<tr>
<td>Valid votes</td>
<td>n.a.</td>
<td>26,908</td>
<td>25,103</td>
<td>25,146</td>
</tr>
<tr>
<td>Threshold 3:</td>
<td></td>
<td>2% per cent of valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valid national</td>
<td>69,189</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Parties' Votes:</td>
<td></td>
<td>30,013</td>
<td>5,513</td>
<td>7,635</td>
</tr>
<tr>
<td>K. Christian People's Party</td>
<td></td>
<td>97,266</td>
<td>40,241</td>
<td>30,358</td>
</tr>
</tbody>
</table>
e-Voting Software

Informal EBON

class_chart LOGICAL_CLOCK

explanation
"A logical clock."

query
"What is the current time of this clock?"

command
"Advance the clock; update the clock's time."

constraint
"The time must be non-negative."
"Must support concurrent use by multiple clients."

end

Listing 1:

In this example, the concepts identified through domain analysis are alarm, alarm clock, and logical clock. Their relationships are summarized in the EBON static diagram CONCEPTS AND RELATIONS in Listing 1. Their definitions are elided in this example.

Each concept is summarized with an informal diagram. An informal diagram describes the concept and its interfaces in terms of queries, commands, and constraints. Queries and commands are collectively known as features.

For example, the logical clock must store a time value and, in EBON terminology, support a query to determine the current time stored in the clock. A command is also necessary to monotonically advance the time stored in the clock. Furthermore, a constraint states that the time stored in the clock is always non-negative. Finally, the logical clock must also behave correctly while being used by multiple concurrent clients.

Formal EBON

Listing 2:

This interface and requirements are expressed using an EBON informal chart. Like most requirement languages, informal EBON uses structured English to denote analysis concepts and requirements. The EBON class chart shown in Listing 2 captures this information.

JML

/**
 * A logical clock.
 * @title "TickTockClock"
 * @date "2007/01/23 18:00:49"
 * @author "Fintan Fairmichael"
 * @organisation "CSI School, UCD"
 * @copyright "Copyright (C) 2007 UCD"
 * @version "$ Revision: 1.7 $"
 */
public interface LogicalClock {

// The current time of this clock.
//@ public model instance _time;
//@ public invariant 0 <= _time;

/**
* @return What is the current time of this clock?
* @concurrency CONCURRENT
*/
//@ ensures _result == _time;
//@ ensures (* _time has been increased. *);
public /*@ pure @*/ long getLogicalTime();

/**
* Advance this clock’s time.
* @concurrency GUARDED
*/
//@ assignable _time;
//@ ensures (* _time has been increased. *);
public void advance();

}

Java

/**
 * A logical clock implementation.
 * @author "Joseph Kiniry"
 */
public class LogicalClockImpl implements LogicalClock {

/**
 * The current time.
 */
private long my_time = 0; //@ in _time;
//@ private represents _time <- my_time;

public long getLogicalTime() {
    return my_time;
}

public void advance() {
    my_time++;
}

}
Refinement Relation
If the input is as we characterized, then we guarantee a correct tally as output.

Overall Correctness Argument

Proof is aggregate modular verification of system’s components.
Governments do not trust Verification
Governments think they trust Testing
Automated Testing that complements Formal Verification
1.2.2. Step Two: Determining of Passing the Threshold
This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of three requirements. Thus, the Danish electoral system has not one, but three different electoral thresholds, and parties qualify for participation in the allocation of compensatory seats by any one of them. The three thresholds are:

1. winning a seat directly in any of the ten multi-member constituencies;
2. obtaining in two of the three electoral provinces a number of votes corresponding - at least - to the provincial votes seat ratio existing in the calculation of these ratios the number of seats in the multi-member constituencies in the electoral provinces in question, excluding the provinces’ compensatory seats; or
3. 2 per cent of the valid, national vote.

For parties that do not meet the first requirement (in 2007 it was two of nine participating parties), the relevant numbers are shown in table 2, which allows a comparison of thresholds (2) and (3), and the votes for the two parties in question in the three electoral provinces as well as nationally.

Experience shows that threshold (3), the 2 per cent rule, is much more important than threshold (2), the vote seat ratio in two of three electoral provinces. Parties that meet the 2 per cent requirement will often also have met threshold (2) - as was the case in 2007 with the Unity List - while parties below the 2 per cent hurdle almost invariably will not meet any of the other requirements (as shown by the example of the Christian Peleers’ Party in 2007, which failed to cross any of the three thresholds).

This experience illustrates how Danish political parties are not (any longer) primarily local or provincial in their support patterns.

1.2.3. Step Three: Allocating Compensatory Seats to Parties
This is the decisive step, since it is here that the proportional, overall, national (or upper tier) allocation of all 175 seats takes place. The calculation (reproduced in table 3 below) allocates the seats available to parties which have qualified for participation in this allocation in strict proportionality to the number of votes obtained by these parties. The calculation is done on the basis of the so-called pure quota; seats not allocated by the full

Table 2. How the Parties that Failed to Qualify for Seats at Threshold (1) Fared on Threshold (2) and (3), November 13, 2007.

<table>
<thead>
<tr>
<th>All of Denmark</th>
<th>Metropolitan Copenhagen</th>
<th>Sealand-Southern Jutland</th>
<th>Northern and Central Jutland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid votes per multi-member constituency seat</td>
<td>n.a.</td>
<td>25,906</td>
<td>25,103</td>
</tr>
<tr>
<td>Threshold 2: 2 per cent of valid national votes</td>
<td>69,189</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>The Parties’ Votes:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K. Christian Peleers’ Party</td>
<td>30,013</td>
<td>5,513</td>
<td>7,635</td>
</tr>
<tr>
<td>Y. New Alliance</td>
<td>97,296</td>
<td>42,241</td>
<td>30,358</td>
</tr>
</tbody>
</table>
Table 1 gives - as an example - the numbers from the multi-member constituency of Jutland (Eastern Jutland).

1.2.2. Step Two: Determining of Passing the Threshold
This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of three requirements. Thus, the Danish electoral system has not one, but three different electoral thresholds, and parties qualify for participation in the allocation of compensatory seats by any one of them. The three thresholds are:

1. winning a seat directly in any of the ten multi-member constituencies;
2. obtaining in two of the three electoral provinces a number of votes corresponding – at least – to the provincial votes seat ratio. Using in the calculation of these ratios the number of seats in the multi-member constituencies in the electoral provinces in question, excluding the provinces’ compensatory seats; or
3. 2 per cent of the valid, national vote.

For parties that do not meet the first requirement (in 2007 it was two of nine participating parties), the relevant numbers are shown in Table 2, which allows a comparison of thresholds (1) and (2), and the votes for the two parties in question in the three electoral provinces as well as nationally.

Experience shows that threshold (3), the 2 per cent rule, is much more important than threshold (2), the vote seat ratio in two of three electoral provinces. Parties that meet the 2 per cent requirement will often also have met threshold (2) — as was the case in 2007 with the Unity List — while parties below the 2 per cent hurdle almost invariably will not meet any of the other requirements (as shown by the example of the Christian People’s Party in 2007, which failed to cross any of the three thresholds). This experience illustrates how Danish political parties are not simply local or provincial in their support patterns.

1.2.3. Step Three: Allocating Compensatory Seats to Parties
This is the decisive step, since it is here that the proportional, overall, national (or upper-tier) allocation of all 175 seats takes place. The calculation (reproduced in Table 3 below) allocates the seats available to parties which have qualified for participation in this allocation in strict proportionality to the number of votes obtained by these parties. The calculation is done on the basis of the so-called pure candidates quota; seats allocated not by the full

Table 2. How the Parties that Failed to Qualify for Seats at Threshold (1) Fared at Threshold (2) and (3), November 13, 2007

<table>
<thead>
<tr>
<th></th>
<th>All of Denmark</th>
<th>Metropolitan</th>
<th>Sealand-Southern Jutland</th>
<th>Northern and Central Jutland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid votes per multi-member constituency seat</td>
<td>n.a.</td>
<td>26,906</td>
<td>25,103</td>
<td>25,146</td>
</tr>
<tr>
<td>Threshold 2: 2 per cent of valid national votes</td>
<td>69,149</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>


---

**Candidate**

| votes: set Ballot, -- First preference ballots assigned to this candidate |
| transfers: set Ballot, -- Second and subsequent preferences received |
| surplus: set Ballot, -- Ballots transferred to another candidate election |
| wasted: set Ballot, -- Ballots non-transferable due to exhaustion of preferences |
| outcome: Event, -- Election result for candidate and associated ballots |

```plaintext
// Non-transferable ballots
0 < #wasted implies outcome = WinnerNonTransferable or outcome = EarlyLoserNonTransferable or outcome = SoreLoserNonTransferable

(outcome = WinnerNonTransferable or outcome = QuotaWinnerNonTransferable)
implies wasted in surplus
(outcome = EarlyLoserNonTransferable or outcome = SoreLoserNonTransferable)
implies wasted in votes + transfers
// Division of ballots into first preferences and transfers
no b: Ballot | b in votes & transfers
// Division of ballots into piles for each candidate
all b: Ballot | b in votes + transfers implies this is b.assignees
// Selection of surplus ballots for re-distribution
surplus in votes + transfers
Election.method = Plurality implies #surplus = 0 and #transfers = 0
0 < #transfers implies Election.method = STV
// Calculation of surplus for PR-STV election
(outcome = Winner and Election.method = STV) or (outcome = SurplusWinner or outcome = WinnerNonTransferable)
implies Scenario quota = #surplus = #votes
(outcome = Winner or outcome = SurplusWinner or outcome = WinnerNonTransferable)
implies #transfers = 0
(outcome = QuotaWinner or outcome = AboveQuotaWinner or outcome = QuotaWinnerNonTransferable)
implies surplus in transfers
(outcome = QuotaWinner or outcome = AboveQuotaWinner or outcome = QuotaWinnerNonTransferable) iff
(this in Scenario, eliminated and not (#{votes + #transfers < Scenario.threshold})
// All non-sore losers are at or above the threshold
outcome = TiedLoser implies Scenario.threshold <= #{votes + #transfers}
```
e-Voting Test Harness
Table 1 gives - as an example - the numbers from the multi-member constituency of Østjylland (Eastern Jutland).

1.2.2. Step Two: Determining of Passing the Threshold
This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of three requirements. Thus, the Danish electoral system has not one, but three different electoral thresholds, and parties qualify for participation in the allocation of compensatory seats by any one of them. The three thresholds are:

1. winning a seat directly in any of the ten multi-member constituencies;
2. obtaining in two of the three electoral provinces a number of votes corresponding - at least 10% of the valid votes in the elections in the multi-member constituencies in the two provinces in question, excluding the provinces' compensatory seats; or
3. 2 per cent of the valid, national vote.

For parties that do not meet the first requirement (in 2007 it was two of nine participating parties),

the relevant numbers are shown in Table 2, which allows a comparison of thresholds (2) and (3), and the votes for the two parties in question in the three electoral provinces as well as nationally.

Experience shows that threshold (3), the 2 per cent rule, is much more important than threshold (2), the vote/seat ratio in two of the three electoral provinces. Parties that meet the 2 per cent requirement will often also have met threshold (2) - as was the case in 2007 with the Unity List - while the parties below the 2 per cent hurdle almost invariably will not meet any of the other requirements (as shown by the example of the Christian People's Party in 2007, which failed to cross any of the thresholds). This experience illustrates how Danish political parties are not (any longer) primarily local or provincial in their support patterns.

1.2.3. Step Three: Allocating Compensatory Seats to Parties
This is the decisive step, since it is here that the proportional, overall, national or upper-tier allocation of all 17% seats takes place. The calculation (reproduced in Table 3 below) allocates the seats available to parties which have qualified for participation in the allocation in strict proportionality to the number of votes obtained by these parties. The calculation is done on the basis of the so-called pure quota, seats not allocated by the full

<table>
<thead>
<tr>
<th>All of Denmark</th>
<th>Metropolitan</th>
<th>Sealand-Southern Jutland</th>
<th>Northern and Central Jutland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid votes per multi-member constituency seat</td>
<td>n.a.</td>
<td>26,906</td>
<td>25,103</td>
</tr>
<tr>
<td>Threshold 2: 2 per cent of valid national votes</td>
<td>69,119</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>The Parties’ Votes:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K. Christian People’s Party</td>
<td>30,013</td>
<td>5,513</td>
<td>7,635</td>
</tr>
<tr>
<td>Y. New Alliance</td>
<td>97,295</td>
<td>44,241</td>
<td>30,358</td>
</tr>
</tbody>
</table>

Tuesday, 8 March 2011
### Refinement Relation

A logical clock is a system that tracks time. It must store a time value, support a query to determine the current time, and allow the time to be advanced. The logical clock must also be non-negative and support concurrent use by multiple clients.

#### Logical Clock

**Class Chart**

- **Explanation**: A logical clock.
- **Query**: What is the current time of this clock?
- **Command**: Advance the clock; update the clock's time.
- **Constraint**: The time must be non-negative.

#### Logical Clock Implementation

- **Informal Definition**
  - Stores a time value.
  - Supports a query to determine the current time.
  - Allows the time to be advanced.
  - Non-negative time.
  - Supports concurrent use.

#### Formal EBON

**Class Chart**

- **Logical Clock**
  - **Feature**
    - `my_time`: Integer
      - **Query**: What is the current time of this clock?
      - **Command**: Advance this clock's time.
    - **Constraint**: The time must be non-negative.

#### JML

**Implementation**

```java
public class LogicalClockImpl implements LogicalClock {
    private long my_time = 0;

    public long getLogicalTime() {
        return my_time;
    }

    public void advance() {
        my_time++;
    }
}
```

**In Informal EBON**

- **Logical Clock**
  - **Feature**
    - `my_time`: Integer
      - **Query**: What is the current time of this clock?
      - **Command**: Advance this clock's time.
    - **Constraint**: The time must be non-negative.

---

**Refinement Relation**

The informal EBON specification of a logical clock is refined into a formally generated implementation. The refinement process involves specifying the logical clock's behavior in a formal requirement language (JML), which is then used to generate the implementation code.

**Informal EBON**

- **Logical Clock**
  - **Feature**
    - `my_time`: Integer
      - **Query**: What is the current time of this clock?
      - **Command**: Advance this clock's time.
    - **Constraint**: The time must be non-negative.

**JML**

```java
public class LogicalClockImpl implements LogicalClock {
    private long my_time = 0;

    public long getLogicalTime() {
        return my_time;
    }

    public void advance() {
        my_time++;
    }
}
```
90% coverage

Unit Testing from Specs
Manual System Testing from Law

90% coverage with only a dozen system tests

Tuesday, 8 March 2011
**System Testing from Law**

for every unique election outcome
A Formal Model of Voting
A Parameterized Formal Model of Several Voting Schemes
Alloy Model
Law-Alloy Refinement
Rigorous System Test Generation
Core Concepts of Elections

candidate  ballot
scenario

event  method
election
Core Concepts

• candidate

• votes (set of ballots)

• transfers (set of ballots)

• surplus (set of ballots)

• outcome (event)

• ballot

• assignees (set of candidates)

• preferences (sequence of candidates)
Core Concepts

- scenario
- losers (set of candidates)
- winners (set of candidates)
- eliminated (set of candidates)
- threshold (integer minimum # of votes to not be a sore loser)
- quota (integer minimum # of votes for an STV or quota winner)
Core Concepts

- event, exactly one of...
  - Winner, QuotaWinner, CompromiseWinner, TiedWinner, TiedLoser, Loser, TiedEarlyLoser, EarlyLoser, TiedSoreLoser, SoreLoser

- election
  - candidates (set of candidates)
  - seats (integer)
  - method (plurality or STV)
  - ballots (integer # of unspoiled ballots)
Generating Scenarios

• goal: generate and characterize every possible non-isomorphism scenario

• election method, # candidates, # seats

• example outcomes
  • WL or WL in two candidate plurality
  • SSSLLLLLWWW with 10 candidates and 1 seat in STV

• scenarios as lemmas
  • “I bet there can’t be an election outcome like this!”
Coupling Systems

- couple Alloy to jUnit
- generate and save system tests in generic format for reuse across implementations
- perform code coverage analysis
- characterize system correctness
- identify suspicious parts of an implementation
Early Results

• first working run two weeks ago
• stopped after several hours to characterize results
• 91% code coverage
• two cases missed in scenario analysis
• zero bugs detected
• expected 2nd run will achieve >99% coverage
Summary of Current Affairs

• formally specified, validated, and verified election tally software systems for US, NL, IE, and DK

• traceable refinement from law—interpreted as concepts, features, and requirements—to specifications, software, and proofs

• automatic verification using ESC/Java2

• automated unit tests with >90% coverage

• manual system tests with >90% coverage

• automated system tests with >99% coverage

• all research and development done in “spare time”
Next Steps

• formal model of elections

• system model that includes people, parties, bureaucrats, government

• trust-by-design

• software engineering in the face of an adversarial customer (gov. and citizens)

• logic-based voting scheme

• couple LFs to implementation
Danish Council for Strategic Research
Programme Commission on
Strategic Growth Technologies

DemTech

Basin (ETHZ)
Ryan (Lux)

Siemens
Aion Assembly

Schürmann
Kiningry
Markussen

Fredericksberg
Aarhus

5 years
17M direct
32M total

Tuesday, 8 March 2011
Thanks to Collaborators

- KOA
- Dermot Cochran, Fintan Fairmichael, Engelbert Hubbers, Alan Morkan, Martijn Oostdijk
- Vótáil
- Dermot Cochran
- DVT
- Dermot Cochran, Ólavur Kjølbro