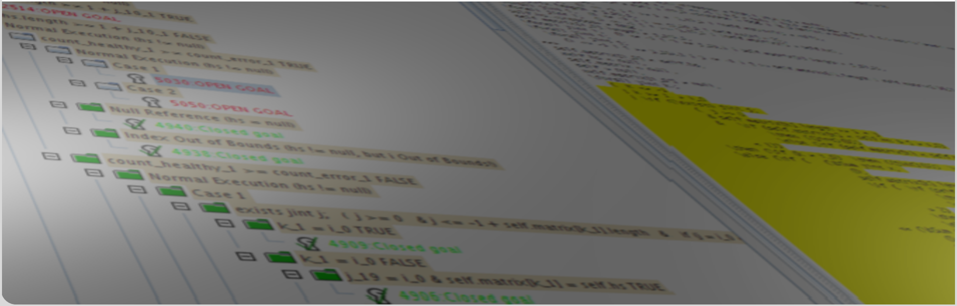
Karlsruhe Institute of Technology

Formal Specification and Verification of Voting Software

Bernhard Beckert | ComSoC, 14.04.13

KARLSRUHE INSTITUTE OF TECHNOLOGY | DEPARTMENT OF COMPUTER SCIENCE



FORMAL SPECIFICATION AND VERIFICATION

Formal Specification and Verification

What?

Logic-based methods for

- specification
  (describing a system's properties)
- verification
  (proving that a system satisfies its specification)

Formal Specification and Verification

Tool Support is Essential

- Automate repetitive tasks
- Avoid clerical errors, etc.
- Cope with large/complex systems
- Make verification certifiable

Formal Specification and Verification

Why?

Dependable Systems

- Safety
- Security

Why?

Dependable Systems

- Safety
- Security



Formal Specification and Verification

Karlsruhe Institute of Technology

Why?

Dependable Systems

- Safety
- Security

Formal Specification and Verification

Why?

Better Understanding of System's Properties

Formal Specification and Verification

Why?

Better Understanding of System's Properties

classical science THEORY EXPERIMENT

Formal Specification and Verification

Why?

Better Understanding of System's Properties

classical science	THEORY	EXPERIMENT
computational science		

Formal Specification and Verification

Why?

Better Understanding of System's Properties

classical science	THEORY	EXPERIMENT
computational science		SIMULATION

Formal Specification and Verification

Why?

Better Understanding of System's Properties

classical science	THEORY	EXPERIMENT
computational science	AUTOMATED REASONING	SIMULATION

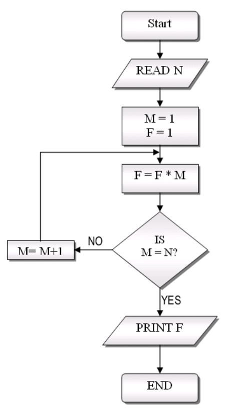
Formal Specification and Verification

Specification may be Declarative or Algorithmic

Formal Specification and Verification

Specification may be Declarative or Algorithmic

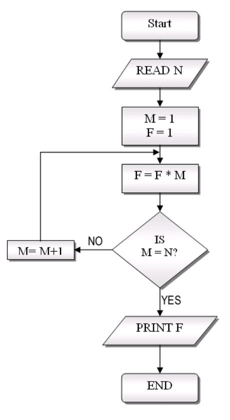
Algorithmic



Formal Specification and Verification

Specification may be Declarative or Algorithmic

Algorithmic



Declarative

$$F = N!$$

Formal Specification and Verification

It is important to know . . .

What System

- Vote casting
- Vote transmission
- Vote counting
- Result verification

What Specification

- Functional
- Security
- Resources
- . . .

What Level of Abstraction

- Declarative description
- Abstract algorithm / flow chart
- Abstract automaton
- Implementation

Formal Specification and Verification

Karlsruhe Institute of Technology

It is important to know . . .

What System

- Vote casting
- Vote transmission
- Vote counting
- Result verification

What Specification

- Functional
- Security
- Resources
- . . .

What Level of Abstraction

- Declarative description
- Abstract algorithm / flow chart
- Abstract automaton
- Implementation

Formal Specification and Verification

Karlsruhe Institute of Technology

It is important to know . . .

What System

- Vote casting
- Vote transmission
- Vote counting
- Result verification

What Specification

- Functional
- Security
- Resources
- . . .

What Level of Abstraction

- Declarative description
- Abstract algorithm / flow chart
- Abstract automaton
- Implementation

VERIFYING INFORMATION-FLOW PROPERTIES

Joint work with

Daniel Bruns, Christoph Scheben, Peter H. Schmitt
Karlsruhe Institute of Technology
(KeY Tool)

Ralf Küsters, Thomas Truderung
University of Trier

Jürgen Graf
Karlsruhe Institute of Technology
(Joanna Tool)

System, Specification, Abstraction Level

System

- Part of simple e-voting system
- Transfer of vote from client to server, computation of result by server

Specification

- Nothing can be learned about votes except the result

Abstraction Level

System: Implementation in Java

Specification: Java Modelling Language

System, Specification, Abstraction Level

System

- Part of simple e-voting system
- Transfer of vote from client to server, computation of result by server

Specification

- Nothing can be learned about votes except the result

Abstraction Level

System: Implementation in Java

Specification: Java Modelling Language

System, Specification, Abstraction Level

System

- Part of simple e-voting system
- Transfer of vote from client to server, computation of result by server

Specification

- Nothing can be learned about votes except the result

Abstraction Level

System: Implementation in Java

Specification: Java Modelling Language

KeY Project

www.key-project.org

Deductive Program Verification

- Java
- Specification:
  Java Modeling Language
- Source-code level

KeY Tool

- Deductive rules for all Java features
- Sequent calculus for Dynamic Logic
- 100% Java Card
- High degree of automation / usability
  >10,000 LOC / expert year

KeY Project

www.key-project.org

Deductive Program Verification

- Java
- Specification:
  Java Modeling Language
- Source-code level

KeY Tool

- Deductive rules for all Java features
- Sequent calculus for Dynamic Logic
- 100% Java Card
- High degree of automation / usability
  $>10,000$ LOC / expert year

KeY Project

www.key-project.org

Deductive Program Verification

- Java
- Specification:
  Java Modeling Language
- Source-code level

KeY Tool

- Deductive rules for all Java features
- Sequent calculus for Dynamic Logic
- 100% Java Card
- High degree of automation / usability
  $>10,000$ LOC / expert year

KeY Project

KeY

www.key-project.org

Deductive Program Verification

- Java
- Specification:
  Java Modeling Language
- Source-code level

KeY Tool

- Deductive rules for all Java features
- Sequent calculus for Dynamic Logic
- 100% Java Card
- High degree of automation / usability
  $>10,000$ LOC / expert year

KeY Project

www.key-project.org

Deductive Program Verification

- Java
- Specification:
  Java Modeling Language
- Source-code level

KeY Tool

- Deductive rules for all Java features
- Sequent calculus for Dynamic Logic
- 100% Java Card
- High degree of automation / usability
  $>10,000$ LOC / expert year

KeY Project

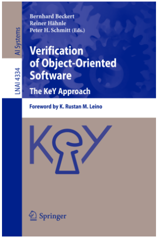
www.key-project.org

Deductive Program Verification

- Java
- Specification:
  Java Modeling Language
- Source-code level

KeY Tool

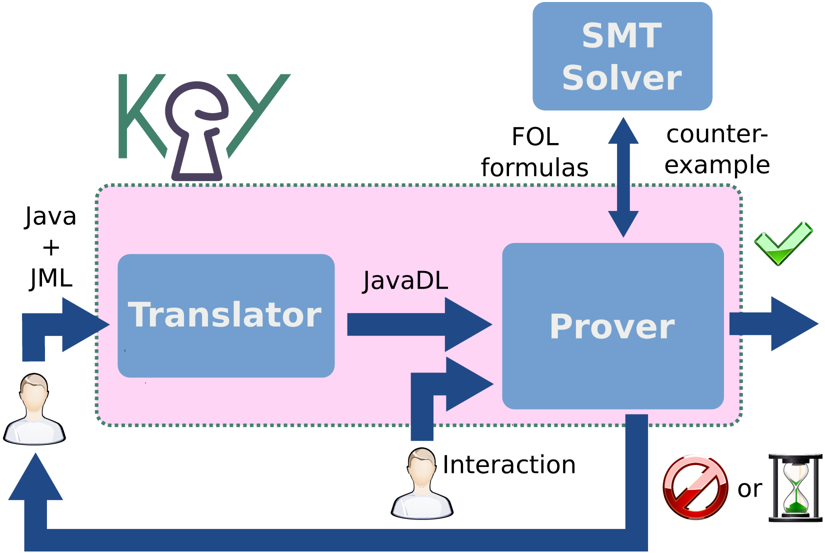
- Deductive rules for all Java features
- Sequent calculus for Dynamic Logic
- 100% Java Card
- High degree of automation / usability
  $>10,000$ LOC / expert year



Example:
JML Specification of a Java Method

```
/*@ requires a.length > 0;
  @ ensures (\forall int i; 0<=i && i<a.length ;
  @             \result <= a[i]);
  @ ensures (\exists int i; 0<=i && i<a.length ;
  @             result == a[i]); @*/
int min(int []a) {
  int i, min; min = a[0];
  /*@ maintaining 0 <= i && i <= a.length;
    @ maintaining (\forall int j; 0 <= j &&
    @      j < i; a[j] >= min);
    @ maintaining (\exists int j; 0 <= j
    @     && j < a.length; min == a[j]); @*/
  for (i = 0; i < a.length; i++)
    { if (a[i] < min) min = a[i]; }
  return min;
}
```

KeY Verification Process



Current State of Project

KIT
Karlsruhe Institute of Technology

Verified

Joanna Tool: No information-flow in communication

Joanna Tool: No information-flow in server besides published result

KeY Tool: Election result correctly computed

KeY Tool: Computed result carries no additional information

Missing

Integrity of votes: Votes not changed during communication

Current State of Project

Verified

Joanna Tool:	No information-flow in communication
Joanna Tool:	No information-flow in server besides published result
KeY Tool:	Election result correctly computed
KeY Tool:	Computed result carries no additional information

Missing

Integrity of votes: Votes not changed during communication

ANALYSING STV VOTING SCHEME USED AT CADE CONFERENCES

Joint work with

Carsten Schürmann
IT University of Copenhagen

Rajeev Goré
Australian National University

System, Specification, Abstraction Level

System

- Single Transferable Vote Algorithm
  as used in election of the CADE Conference board of trustees

Specification

- Properties of election result

Abstraction Level

System: Abstract algorithm formalised in linear logic program (Celf System)

Specification: Axioms formalised in first-order logic

System, Specification, Abstraction Level

System

- Single Transferable Vote Algorithm
  as used in election of the CADE Conference board of trustees

Specification

- Properties of election result

Abstraction Level

System: Abstract algorithm formalised in linear logic program
(Celf System)

Specification: Axioms formalised in first-order logic

System, Specification, Abstraction Level

System

- Single Transferable Vote Algorithm
  as used in election of the CADE Conference board of trustees

Specification

- Properties of election result

Abstraction Level

System: Abstract algorithm formalised in linear logic program (Celf System)

Specification: Axioms formalised in first-order logic

Single Transferable Vote

System for Preferential Voting

- Used in real-world elections
- Proportional representation
- Does not necessarily elect Condorcet winner

Single Transferable Vote

"Standard" Version

Quota $Q := \left\lfloor \frac{votes}{seats+1} \right\rfloor + 1$

Repeat until all seats filled (or not enough candidates left)

- if candidate with Q first-preference votes exists:

  declare elected

  delete Q of the votes

  delete from ballot-box
- else

  delete weakest candidate from ballot-box

Various choice points! Various versions!

Single Transferable Vote

"Standard" Version

Quota $Q := \left\lfloor \frac{votes}{seats+1} \right\rfloor + 1$

Repeat until all seats filled (or not enough candidates left)

- if candidate with Q first-preference votes exists:
  - declare elected
  - delete Q of the votes
  - delete from ballot-box
- else
  - delete weakest candidate from ballot-box

Various choice points! **Various versions!**

Example

Candidates: A, B, C, D

Seats: 2

Votes:

$$A > B > D$$
$$A > B > D$$
$$A > B > D$$
$$D > C$$
$$C > D$$

Example

Candidates: A, B, C, D

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

Seats: 2

Votes:

$$A > B > D$$
$$A > B > D$$
$$A > B > D$$
$$D > C$$
$$C > D$$

Example

Candidates: $A, \quad B, \quad C, \quad D$ $\qquad$ $Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2$

Seats: 2

Votes:

$$A > B > D \quad 1$$
$$A > B > D \quad 2$$
$$A > B > D \quad 3$$
$$D > C$$
$$C > D$$

Example

Candidates: $A, \quad B, \quad C, \quad D$ $\qquad Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2$

Seats: 2

Votes:

$$A > B > D \quad 1$$
$$A > B > D \quad 2$$
$$A > B > D \quad 3$$
$$D > C$$
$$C > D$$

Elected: A

Example

Candidates: A, B, C, D

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

Seats: 2

Votes:

~~$A > B > D$~~ 1
~~$A > B > D$~~ 2
$A > B > D$ 3
$D > C$
$C > D$

Elected: A

Example

Candidates: $A, \quad B, \quad C, \quad D$

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

Seats: 2

Votes:

~~$A > B > D$~~

~~$A > B > D$~~

~~A~~ $> B > D$

$D > C$

$C > D$

Elected: A

Example

Candidates: A, B, C, D

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

Seats: 2

Votes:

~~$A > B > D$~~

~~$A > B > D$~~

~~A~~ $>$ ~~B~~ $> D$

$D > C$

$C > D$

Elected: A

Example

Candidates: A, B, C, D

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

Seats: 2

Votes:

~~$A > B > D$~~
~~$A > B > D$~~
~~A~~ $>$ ~~B~~ $> D$ 1
$D > C$ 2
$C > D$

Elected: A

Example

Candidates: $A, \quad B, \quad C, \quad D$ $\qquad Q = \left\lfloor \frac{5}{2+1} \right\rfloor + 1 = 2$

Seats: 2

Votes:

~~$A > B > D$~~

~~$A > B > D$~~

~~A~~ $>$ ~~B~~ $> D$ 1

$D > C$ 2

$C > D$

Elected: $A, \quad D$

Declarative Description

Computes an approximation to an optimisation problem

therefore IMPOSSIBLE in PRACTICE

Precise functional specification covering all inputs

Declarative Description

Computes an approximation to an optimisation problem

therefore IMPOSSIBLE in PRACTICE

Precise functional specification covering all inputs

Declarative Description

Two Properties

- There are enough votes for each elected candidate
  (ignoring preferences)
- Election result is consistent with union U of preferences
  if U is consistent
  (ignoring number of votes)

Declarative Description

Formalisation of 1st Property

$$
\begin{array}{l}
\exists a(\\
\quad \forall i(1 \leq i \leq \mathtt{V} \rightarrow 0 \leq a[i] \leq \mathtt{S}) \wedge \\
\quad \forall i(1 \leq i \leq \mathtt{V} \rightarrow (a[i] \neq 0 \rightarrow r[a[i]] \neq 0) \wedge \\
\quad \forall i((1 \leq i \leq \mathtt{V} \wedge a[i] \neq 0) \rightarrow \exists j(1 \leq j \leq \mathtt{C} \wedge b[i,j] = r[a[i]])) \wedge \\
\quad \forall k((1 \leq k \leq \mathtt{S} \wedge r[k] \neq 0) \rightarrow \\
\qquad \exists count(count[0] = 0 \wedge \\
\qquad\qquad \forall i(1 \leq i \leq \mathtt{V} \rightarrow (a[i] = k \rightarrow count[i] = count[i-1] + 1) \wedge \\
\qquad\qquad\qquad (a[i] \neq k \rightarrow count[i] = count[i-1])) \wedge \\
\qquad\qquad count[\mathtt{V}] = \mathtt{Q})) \\
\quad)
\end{array}
$$

Bounded Model Checking

[Beckert/Goré/Schürmann, CADE 2013]

Method

- Generate all possible ballot-boxes (up to certain bounds)
- Run through algorithm implemented in linear logic program (Celf)
- Check result w.r.t. properties

Single Transferable Vote @CADE

Quote from CADE Bylaws (legal document)

```
Procedure STV

Elected <-- empty
T <-- Tbl              {* Start with the original vote matrix *}
for E <-- 1 to K
    N' <-- N-E+1   {* Choose a winner among N' candidates *}
    T' <-- T       {* store the current vote matrix *}
    while (no candidate has a majority of 1st preferences)
        w <-- one weakest candidate
        for all candidates c {* remove all weakest candidates *}
            if c is equally weak as w
                Redistribute(c,T)
        end for
    end while
    win <-- the majority candidate
    Elected <-- append(Elected, [win])
    T <-- T'       {* restore back to N' candidates *}
    Redistribute(win, T)  {* remove winner & redistrb. votes *}
end for

End STV
```

Differences CADE-STV / Standard STV

CADE-STV

- Quota: $>$50% of votes (majority)
- Restart with original ballot-box
  (deleted votes and weakest candidates come back)
- No autofill if not enough candidates

Example

Candidates: $A, \quad B, \quad C, \quad D$

Seats: 2

Votes:

$$A > B > D$$
$$A > B > D$$
$$A > B > D$$
$$D > C$$
$$C > D$$

Example

Candidates: $A, \quad B, \quad C, \quad D$ $\qquad Q = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3$

Seats: 2

Votes:

$$A > B > D$$
$$A > B > D$$
$$A > B > D$$
$$D > C$$
$$C > D$$

Example

Candidates: $A, \quad B, \quad C, \quad D$ $\qquad Q = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3$

Seats: 2

Votes:

$$A > B > D \quad 1$$
$$A > B > D \quad 2$$
$$A > B > D \quad 3$$
$$D > C$$
$$C > D$$

Example

Candidates: $A, \quad B, \quad C, \quad D$ $\qquad Q = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3$

Seats: 2

Votes:

$$A > B > D \quad 1$$
$$A > B > D \quad 2$$
$$A > B > D \quad 3$$
$$D > C$$
$$C > D$$

Elected: A

Example

Candidates: A, B, C, D $Q = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3$

Seats: 2

Votes:

$$\not{A} > B > D \quad 1$$
$$\not{A} > B > D \quad 2$$
$$\not{A} > B > D \quad 3$$
$$D > C$$
$$C > D$$

Elected: A

Example

Candidates: A, B, C, D $\quad Q = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3$

Seats: 2

Votes:

~~A~~ $> B > D$

~~A~~ $> B > D$

~~A~~ $> B > D$

$D > C$

$C > D$

Elected: A

Example

Candidates: A, B, C, D $\qquad Q = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3$

Seats: 2

Votes:

$$\cancel{A} > B > D \quad 1$$
$$\cancel{A} > B > D \quad 2$$
$$\cancel{A} > B > D \quad 3$$
$$D > C$$
$$C > D$$

Elected: A

Example

Candidates: $A, \quad B, \quad C, \quad D$ $\qquad Q = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3$

Seats: 2

Votes:

$\not{A} > B > D$ 1

$\not{A} > B > D$ 2

$\not{A} > B > D$ 3

$D > C$

$C > D$

Elected: $A, \quad B$

Example

Candidates: A, B, C, D $\qquad Q = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3$

Seats: 2

Votes:

$$\not{A} > B > D \quad 1$$
$$\not{A} > B > D \quad 2$$
$$\not{A} > B > D \quad 3$$
$$D > C$$
$$C > D$$

Elected: A, B

No proportional representation!
Majority rules!

Conclusions

Karlsruhe Institute of Technology

Conclusion I

Support in reasoning about voting schemes needed

Conclusion II

Can be automated with bounded model checking

Conclusion III

Tailor-made properties for specific voting systems needed

Conclusions

Conclusion I

Support in reasoning about voting schemes needed

Conclusion II

Can be automated with bounded model checking

Conclusion III

Tailor-made properties for specific voting systems needed

Conclusions

Karlsruhe Institute of Technology

Conclusion I

Support in reasoning about voting schemes needed

Conclusion II

Can be automated with bounded model checking

Conclusion III

Tailor-made properties for specific voting systems needed