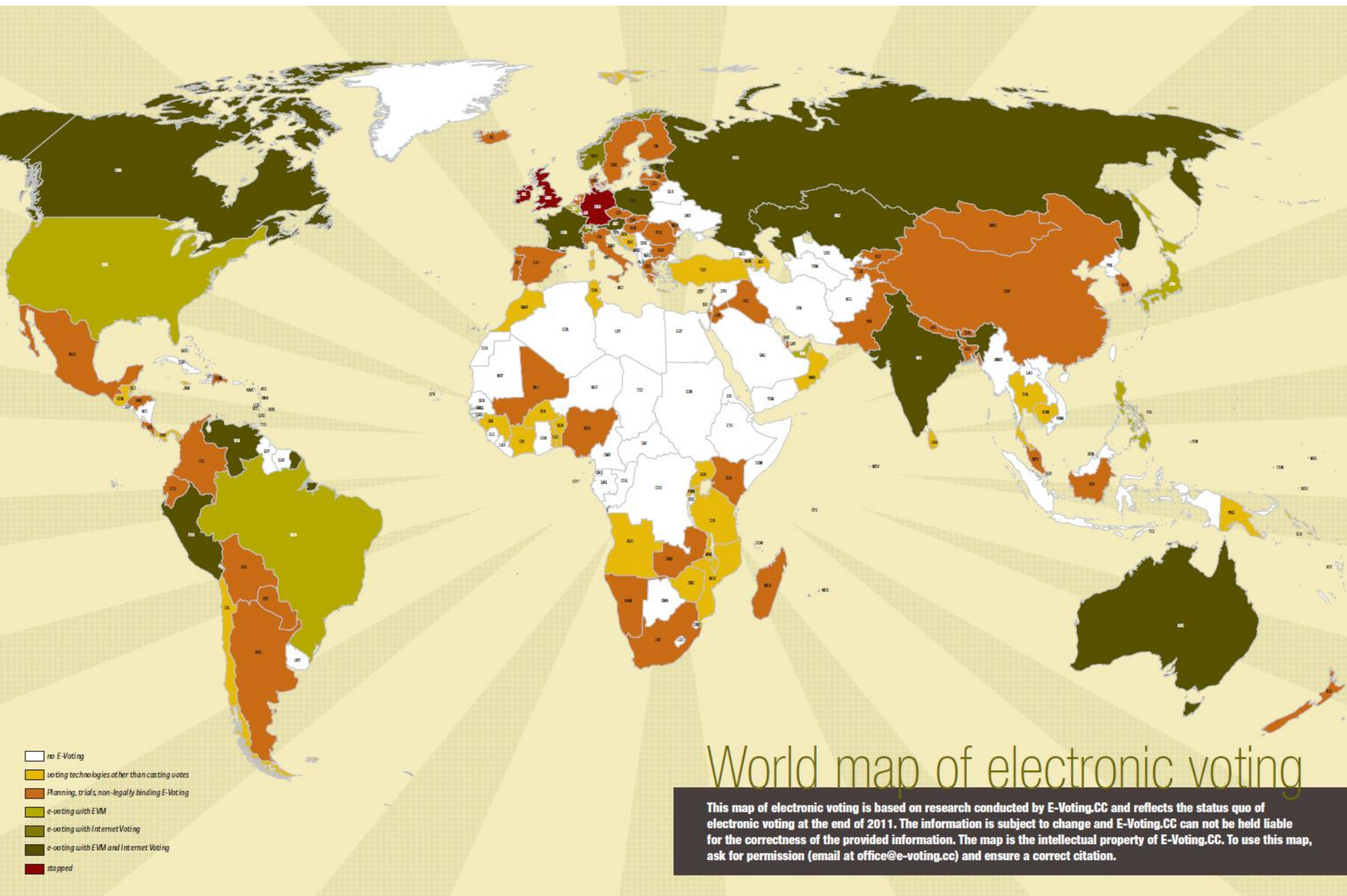


Bounded Verification of Voting Software

Greg Dennis, Kuat Yessenov, Daniel Jackson

Tobias Hartmann



Electronic voting machines

- Used in the Netherlands since 1998
- Introduced KOA Remote Voting System in 2004
 - Internet voting application
 - Java, Open-Source (GPL)
 - Main part implemented by LogicaCMG
- Security of Systems (SoS) research group developed independent tally subsystem

The Problem

- How to verify that the software is correct?
 - Electoral systems are highly complex
- Vote-tallying subsystem of KOA
 - Formally specified using Java Modeling Language (JML)
 - Core partially verified using Extended Static Checker for Java, ESC/Java2 (47%)
 - Boundary unit tests generated using jmlunit (8000)
 - 100% coverage not possible in timeframe of project (4 weeks)

Bounded verification

- Examine all executions of a procedure with bounded
 - Heap size
 - Number of loop unrollings
- Under-approximation
 - Will find counterexample if in bounds
 - Will always miss bugs that require larger bounds
- Relies on small-scope hypothesis
 - Many defects have small counterexamples

Original approach

Java Code + Specification (JML)

Relational first order logic
(Alloy)

Alloy Analyzer (SAT with
external solver)

Forge Intermediate Representation (FIR)

- Simple relational programming language
 - Supports modularity
- Automatic Java-to-FIR translation
 - Relational view of the heap
 - Types as sets
 - Fields as functional relations
 - Local variables as singleton sets
- Basically 3 tools: *Forge*, *JMLForge* and *JForge*

The Forge framework

- From FIR procedure tool obtains
 - Constraint between pre-state s and post-state s' : $P(s, s')$
 - User provided specification $S(s, s')$
- Combined to $P(s, s') \wedge \neg S(s, s')$
 - True for executions that are possible but violate specification
- Bounds on
 - Number of loop unrollings
 - Bitwidth of FIR integers
 - Scope for each domain

New approach: Three stage translation

Java Code + Specification (JML)

Forge Intermediate Representation (FIR)

Kodkod relational logic

Kodkod model finder (SAT with external solver)

Results

- Tested 169 methods of 8 classes
 - Scope of 5, bitwidth of 4 (-8 to 7), 3 loop unrollings
- 19 specification violations found
 - Overspecification, Underspecification or Bug
 - No false alarms
- Minimum bound for each violation to be detected
 - Scope 2, bitwidth 3, 3 loop unrollings
 - Supports small-scope hypothesis

Example violation 1

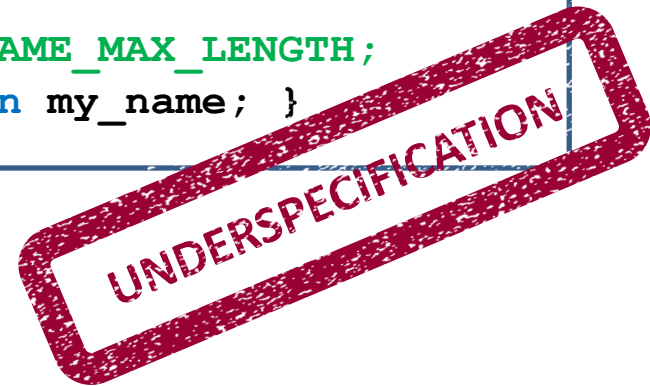
```
class KiesLijst {
    public int compareTo(final Object an_object) {
        if (!(an_object instanceof KiesKring)) {
            throw new ClassCastException();
        }
        final KiesKring k = (KiesKring) an_object;
        return number() - k.number();
    }
}
```



- Unit-Testing did not catch the bug because parameter is of type **Object** instead of **KiesLijst**

Example violation 2

```
//@ requires a_kieskring_name.length() <= KIESKRING_NAME_MAX;  
//@ ensures number() == a_kieskring_number;  
//@ ensures name().equals(a_kieskring_name);  
private /*@ pure @*/ KiesKring(final byte a_kieskring_numer,  
    final /*@ non_null @*/ String a_kieskring_name) {  
    my_number = a_kieskring_number;  
    my_name = a_kieskring_name;  
}  
  
//@ ensures \result.length() <= KIESKRING_NAME_MAX_LENGTH;  
/*@ pure non_null @*/ String name() { return my_name; }
```



- Again missed by Unit-Testing.

Limitations

- Translation from FIR to relational logic
 - Sound
 - Complete within bounds
- Translation from Java to FIR
 - Not all Java statements supported and optimizations introduce imprecision
 - Spurious counterexamples: Integer overflow due to limited bitwidth
 - Missed counterexamples: No real number arithmetic

Conclusion and future work

- Despite a *verification-centric methodology* 19 out of 169 methods violate specification
- Benefits compared to unit testing
- Future improvement of performance necessary
- JMLForge not actively supported anymore, use JForge

References

- Not very detailed and self-contained, had to read other papers as well
 - Greg Dennis, Felix Chang, Daniel Jackson. *Modular Verification of Code with SAT*
 - Joseph R. Kiniry, Alan E. Morkan, Dermot Cochran, Fintan Fairmichael, Patrice Chalin, Martijn Oostdijk, Engelbert Hubbers. *The KOA Remote Voting System: A Summary of Work to Date*
 - Divya Gopinath *Scaling Scope Bounded Checking using Incremental Approaches*
 - Kuat T. Yessenov *A lightweight specification language for bounded program verification*

FIR example

```
class Birthday {
    /*@ non_null */ Month month;
    int day;

    /*@ requires this.month.checkDay(d);
    /*@ ensures this.day == d;
    void setDay(int d) {
        Month m = this.month;
        boolean dayOk = m.checkDay(d);
        if (dayOk) this.day = d;
    }
}

class Month {
    int maxDay;
    /*@ ensures \result <==> (d > 0 && d <= maxDay);
    /*@ pure */ boolean checkDay(int d) { ... }
}
```

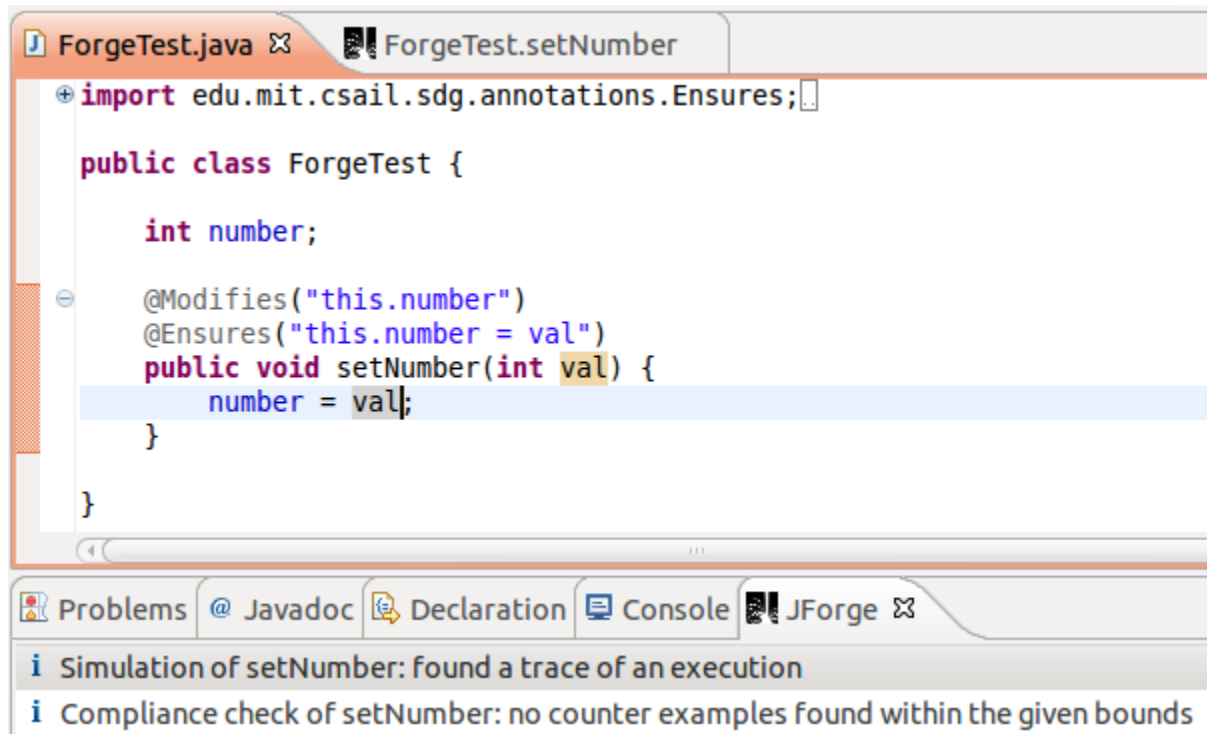

FIR example

```
domain Birthday, domain Month, domain Object
global month: Birthday -> Month
global day: Birthday -> Integer
global maxDay: Month -> Integer
local this: Birthday, local d: Integer
local m: Month, local dayOk: Boolean

proc setDay (this, d): ()
  m = this.month;
  dayOk = spec (dayOk ⇔ (d > 0 AND d <= m.maxDay));
  if dayOk then day = day ⊕ (this -> d) else exit;
```

JForge Example

- Eclipse Plugin, <http://sdg.csail.mit.edu/forge/>
- Uses JForge Specification Language (JFSL)



```
ForgeTest.java  ForgeTest.setNumber
+ import edu.mit.csail.sdg.annotations.Ensures;

public class ForgeTest {

    int number;

    @Modifies("this.number")
    @Ensures("this.number = val")
    public void setNumber(int val) {
        number = val;
    }

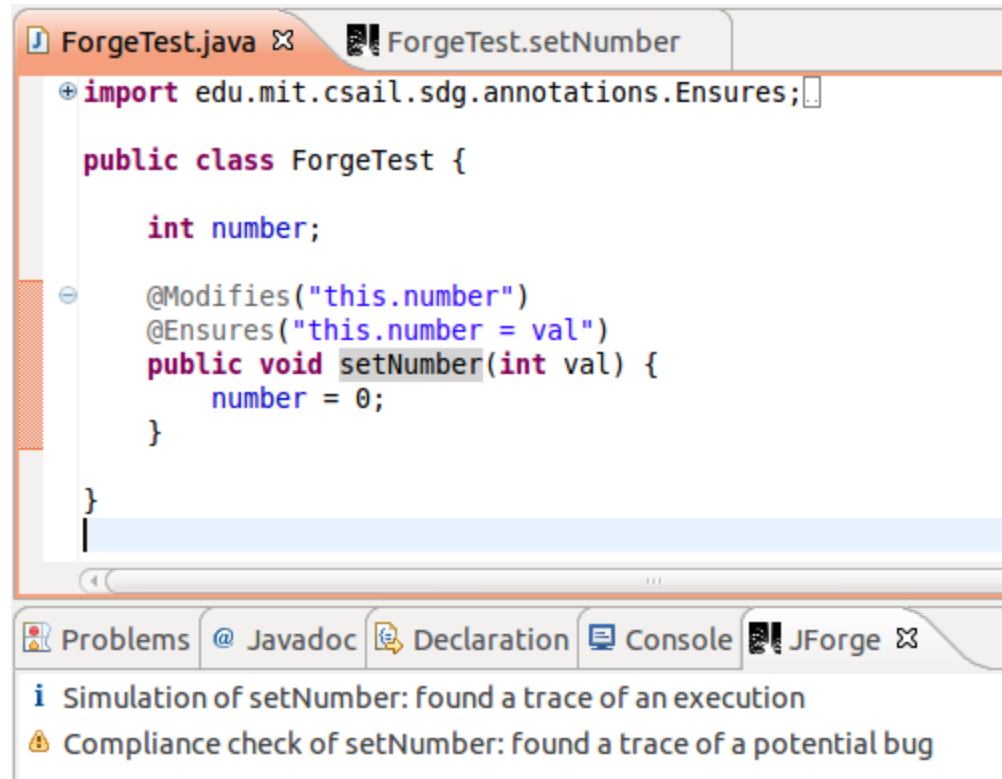
}
```

Problems @ Javadoc Declaration Console JForge

i Simulation of setNumber: found a trace of an execution

i Compliance check of setNumber: no counter examples found within the given bounds

JForge Example



```
ForgeTest.java ✕ ForgeTest.setNumber
+ import edu.mit.csail.sdg.annotations.Ensures;

public class ForgeTest {

    int number;

    @Modifies("this.number")
    @Ensures("this.number = val")
    public void setNumber(int val) {
        number = 0;
    }

}
```

Problems @ Javadoc Declaration Console JForge ✕

- i Simulation of setNumber: found a trace of an execution
- ⚠ Compliance check of setNumber: found a trace of a potential bug

JForge Example

The screenshot shows an IDE window with two tabs: "ForgeTest.java" and "ForgeTest.setNumber". Below the tabs, the text "Trace of the execution" is displayed in orange. A blue link "Open trace visualizer in a separate window" is present. Below this, a list of trace items is shown:

- Root cause #1:
`((this.number) = val)`
- Trace evaluator:
- Trace visualizer:

The trace visualizer shows a diagram with three nodes:

- A root node: `ForgeTest0: this`
- A left child node: `0` (connected to the root by a red arrow labeled "number")
- A right child node: `-4: val` (connected to the root by a black arrow labeled "number")

To the right of the diagram is a separate box containing the text: `null: setNumber throw`

Below the diagram, the text "Complete trace:" is visible.