Pflichtenheft

Beste Gruppe

22. November 2016

CBMC C Bounded Model Checker

BMC Bounded Model Checking

GUI Graphical User Interface

Inhaltsverzeichnis

1 Produktübersicht

Wahlverfahren bilden den Grundstein unserer Demokratie. Dabei werden viele Anforderungen an sie gestellt, welche unsere intuitiven Ideen über Gerechtigkeit formalisieren: Proportionalität, Anonymität, etc. Moderne Wahlverfahren sind oft so komplex, dass sie viele überraschende und teils unerwünschte Eigenschaften haben. Nachweisen deren Abwesenheit ist absolut nicht trivial. So wurde beispielsweise 2008 das Bundestagswahlrecht vom BVerfG für verfassungswidrig erklärt, da es unter anderem Gleichheit der Wirkung verschiedener Stimmen verletzte. Auf der anderen Seite ist es auch sehr schwer, Wahlverfahren auf die Anwesenheit erwünschter Eigenschaften zu untersuchen.

Bounded Model Checking (BMC) bietet im allgemeinstem Fall eine Möglichkeit zu verifizieren, ob ein gegebenes System diverse Eigenschaften erfüllt. Es ist vollautomatisch und gibt bei Nichterfüllung ein Gegenbeispiel zurück. Dazu baut es eine boolsche Formel, welche genau dann erfüllbar ist, wenn die zu Untersuchende Eigenschaft vom System nicht erfüllt wird.

In unserem Fall kann BMC konkret dazu verwendet werden, ein C-Programm darauf zu untersuchen ob es im Falle gegebener Vorbedingungen gegebene Nachbedingungen erfüllt. Dies wird dazu verwendet, obige Problemstellung zu lösen: so kann ein in C definiertes Wahlverfahren wie z.B. die einfache Mehrheitswahl darauf geprüft werden, ob es bestimmte Eigenschaften erfüllt. Allerdings ist es sehr kompliziert, dies ohne Weiteres zu tun.

Unser Programm ist im Wesentlichen eine eine sehr umfangreiche Schnittstelle um mit C Bounded Model Checker (CBMC) zu kommunizieren. Es bietet dem Benutzer über eine Graphical User Interface (GUI) die Möglichkeiten, formale Eigenschaften für Wahlverfahren sowie diese Wahlverfahren selbst anzugeben und zu editieren. Weiterhin liefert es Möglichkeiten die Interaktion mit CBMC zu gestalten: Für wie viele Wähler, Plätze etc geprüft werden soll. Nach erfolgreicher Überprüfung durch CBMC bekommt der Benutzer schließlich eine Antwort des Programms, in der er bei Nichterfüllung der Eigenschaft ein Gegenbeispiel angezeigt bekommt. Sollte die Prüfung jedoch erfolgreich sein, bekommt der Nutzer ein Positivbeispiel präsentiert. All dies wird graphisch über die GUI aufbereitet.

Die GUI ist in vier Teilen angeordnet:

1. „Rigtime“: Code-Editor für Wahlverfahren in Programmiersprache C.

2. „Properties“: Editor für Spezifikation formaler Eigenschaften in abgespeckter C-Syntax mit speziellen Macros.

3. „Params“: Eingabe von Parametern einer zu analysierenden Wahl mit Anzahl der Wähler, Kandidaten und Sitzen. s

4. „Rigplete“: Ausgabe der Prüfung.

2 Zielbestimmung

Ziel des Programmes ist es, eine komfortable, graphische Lösung zur Untersuchung formaler Eigenschaften von Wahlverfahren zu präsentieren, welche auch von Nicht-Informatikern mit minimalem Aufwand erlernt und eingesetzt werden kann. Es soll Folgendes bereitstellen:

- Eine Möglichkeit zur Beschreibung diverser Wahlverfahren in C-Code
- Eine Möglichkeit zur Beschreibung der formalen Eigenschaften, welche das Wahlverfahren erfüllen sollen, in abgespeckter C Syntax
- Eine Möglichkeit zum Angeben der Eingabewerte (Anzahl Wähler, Anzahl Kandidaten, Anzahl Sitze)
- Eine Ausgabe des Ergebnisses der Überprüfung: eine Erfolgsmeldung bei Erfolg und Präsentation eines Gegenbeispiels bei Nichterfolg

Die letztendliche Überprüfung wird durch den CBMC geschehen. Aufgabe des Programmes wird es sein, die gegebenen Eingaben für den CBMC aufzubereiten sowie dessen Ausgabe zu interpretieren und präsentieren.

All diese Aufgaben ließen sich theoretisch auch schon jetzt, ohne Verwendung unseres Programms erledigen. Allerdings wäre der damit verbundene Lern- und reguläre Aufwand sehr hoch, vor allem bei der Angabe der formalen Eigenschaften. Daher ist enorm wichtiger Schwerpunkt unseres Programmes einfache Benutzung, auch und besonders für Nicht-Informatiker. Dies soll erreicht werden über eine leicht zu navigierende und intuitive GUI. Dadurch soll das Untersuchen von Wahlverfahren deutlich leichter und schneller werden, was den Mehrwert unseres Programmes ausmacht.

2.1 Musskriterien

- Das Programm kann auf aktuellen Versionen von Windows und Linux-Betriebssystemen betrieben werden.
- Alle Abhängigkeiten (u.a. zu CBMC) werden mit dem Programm ausgeliefert.

Die Elemente der GUI sollen folgenden Kriterien genügen:

- „Rigtime“: Code-Editor mit der Möglichkeit zum Speichern, Speichern unter und Laden.

- „Properties“: Editor mit der Möglichkeit zum Speichern, Speichern unter und Laden. Die Eingabe soll überprüft werden, d.h. ob die Eingabe Macros in C-Syntax darstellt. Im selben Fenster soll es eine Eingabemaske für zusätzliche symbolische Variablen geben.
- „Params“: Option für eine graphische Eingabe der Anzahl von Wählern, Kandidaten und Sizen.
- „Rigplete“: Das Ergebnis der Wahlanalyse wird angezeigt. Falls CBMC ein Gegenbeispiel zu einer formalen Eigenschaft gefunden hat, soll das Beispiel XXX graphisch XXX angezeigt werden.

2.2 Wunschkriterien

- Das Programm kann auf aktuellen Macs betrieben werden.

Die Elemente der GUI können folgenden Kriterien genügen:

- „Rigtime“: Der Code-Editor soll für die Programmiersprache C bieten:
  - Syntax-Highlighting
  - Fehler-Anzeige
  - Automatisches Einrücken
  - Auto completion
  - Widerrufen, Wiederherstellen
  - Tastatur-Shortcuts
  - Warnung vor nicht unterstützten Elementen der Programmiersprache
  - Codevorlagen
- „Properties“: Der Editor soll Syntax-Highlighting und eine Fehler-Anzeige für C bieten. Codeeingaben sollen komplettiert werden können.
- „Params“: Die Analyse der Wahl soll mit Hilfe eines Buttons abgebrochen werden können.
- Wahlergebnisausgabe: Im Fenster XXX „Params“ XXX kann ein Array mit Wahlstimmen eingegeben werden. Das Ergebnis dieser Wahl wird im Fenster „Rigplete“ angezeigt.
- Zusätzlicher SAT-Solver: Das Programm bietet eine Schnittstelle, sodass auch andere SAT-Solver als CBMC angesteuert werden können.
- XXX Weitere? XXX

2.3 Abgrenzungskriterien

XXX

3 Produkteinsatz

Das Programm überprüft Wahlverfahren auf ihre formalen Eigenschaften. Es richtet sich an Kunden, die ein Interesse an der Erforschung oder Entwicklung solcher Verfahren haben. Sie benötigen ein Grundverständnis der Programmiersprache C und Logik.

3.1 Anwendungsbereiche

- Universitärer Bereich
- Forschung

3.2 Zielgruppen

- Wahlforscher
- Softwareentwickler

3.3 Betriebsbedingungen

XXX welche gibt es? XXX

4 Produktumgebung

4.1 Software

- OS: Windows/Linux
- Softwareentwickler

4.2 Hardware

- PC

4.3 Orgware

XXX

4.4 Produkt-Schnittstellen

XXX bei Implementierung von Einsatz anderer SAT-Solver? XXX XXX allgemein: Ausgabe von Produktdaten? XXX

5 Funktionale Anforderungen

5.1 Allgemein

/F10/ Bereitstellen von Editoren zur Beschreibung des Wahlverfahrens sowie zu erfüllender formaler Eigenschaften
/F20/ Kommunikation und Überprüfung dieser Eigenschaften via CBMC
/F30/ Bereitstellen von Kommunikationsschnittstellen mit CBMC sowohl für Eingabe von Parametern als auch Ausgabe der Ergebnisse, welche auch für Nicht-Informatiker verständlich ist
/F40/ Möglichkeit des Speicherns von Code, formaler Anforderungen und Eingabeparametern als ein Projekt

5.2 C-Code Editor für Wahlverfahren

/F10/ Darstellung aller für das Programmieren in C benötigten Charaktere
/F20/ Veränderung des dargestellten Textes durch Eingabe anderer Charaktere über die Tastatur wie in Notepad
/F30/ Speichern von erstellten Code in Dateiformat datei.c
/F40/ Laden und Darstellen von .c Dateien
/F50/ Automatisches Einrücken des Codes in Schleifen und if-Statements
/F60/ Code-Completion

- Automatisches Schließen von Klammern und Anführungszeichen
- Primitiv: Vorschlagen bereits im Code vorgekommener Wörter
- Intelligent: Durch Analysieren eines ASTs nur Vorschlagen der Wörter welche im Kontext Sinn ergeben.

/F70/ Syntax-Highlighting: Darstellung diverser Schlüsselwörter in anderen Farben als den Rest des Codes. Dies beinhaltet, ist jedoch nicht beschränkt auf:

- Typendeklaration (int, float, structs...)
- Kontrollflow-Konstrukte (if, else, while...)
- Kommentare

/F80/ Durch den User konfigurierbares Verhalten:

Tabelle 5.1: Hotkeys und verbundene Operationen

Kürzel	Operation
Strg + c	Kopieren
Strg + x	Auschneiden
Strg + v	Einfügen
Strg + z	Zuletzt ausgeführte Aktion Rückgängig machen
Strg + r	Zuletzt rückgängig gemachte Aktion erneut ausführen
Strg + s	Speichern
Strg + o	Öffnen
Strg + Leer	Anzeigen der Code-Completion Vorschläge

- Festlegen der Farben, welche beim Syntax-Highlighting verwendet werden
- Festlegen des verwendeten Fonts

/F90/ Anzeigen von Syntaktischen Fehlern im Code, welche durch einen Lexer oder Parser erkannt werden können:

- Verwendung von Schlüsselwörtern als Variablennamen
- Vergessene Semikolons am Ende von Anweisungen

/F100/ Reaktion auf typische Tastenkürzel (siehe 5.1)

/F110/ Bereitstellen von Wahl-Templates

- Jeder Wähler wählt genau einen Kandidaten
- Jeder Wähler ordnet Kandidaten nach Präferenz in absteigender Reihenfolge
- Jeder Wähler ordnet Kandidaten eine Nummer zwischen 100 (maximale Zustimmung) und 0 (maximale Abneigung) zu

5.3 Editor für formale Eigenschaften

/F10/ Darstellung aller für das Programmieren in C benötigten Charaktere
/F20/ Veränderung des dargestellten Textes durch Eingabe anderer Charaktere über die Tastatur wie in Notepad
/F21/ Beschreibung formaler Eigenschaften als Vor- und Nachbedingung in abgespeckter C-Syntax
/F30/ Bereitstellung von Makros zur Beschreibung der Eigenschaften (siehe 5.2)

/F40/ Bereitstellen symbolischer Variablen für Wähler, Kandidaten und Sitze
/F50/ Bereitstellen von Operatoren für Implikation und Äquivalenz
/F60/ Beliebig tiefe, lediglich von Hardware begrenzte, Schachtelung dieser Konstrukte

Tabelle 5.2: Makros zur Beschreibung formaler Eigenschaften

Makro	Effekt
`FOR_ALL_VOTERS(E)`	checkt ob E für alle Wähler gilt
`FOR_ALL_CANDIDATES(E)`	checkt ob E für alle Kandidaten gilt
`FOR_ALL_SEATS(E)`	checkt ob E für alle Sitze gilt
`EXISTS_ONE_VOTER(E)`	checkt ob E für zumindest einen Wähler gilt
`EXISTS_ONE_CANDIDATE(E)`	checkt ob E für zumindest einen Kandidaten gilt
`EXISTS_ONE_SEAT(E)`	checkt ob E für zumindest einen Sitz gilt
`VOTE_SUM_FOR_CANDIDATE(c)`	gibt die Anzahl Stimmen für Kandidaten c zurück

/F70/ Syntax-Highlighting
/F80/ Anzeigen von Syntaktischen Fehlern im Code
/F90/ Code-Completion

- Auto-Vervollständigung der Makros
- Analyse des Codes und Anzeigen relevanter, bereits definierter Eigenschaften und symbolischer Variablen

5.4 Editor für Eingabeparameter

/F10/ Möglichkeit zur Angabe der zu analysierenden Anzahl von Wählern, Kandidaten und Sitzen
/F20/ Möglichkeit zum Eingeben einer Zeitspanne nach welcher die Berechnung abgebrochen wird

5.5 Ausgabe der Analyseergebnisse

/F10/ Ausgabe einer Erfolgsmeldung bei Erfolg
/F20/ Graphische Darstellung eines Gegenbeispiels

6 Produktdaten

6.1 Code-Editor „Rigtime“

/D10/ Das Wahlverfahren ist als Methode „unsigned int voting(params)“ einer C-Headerdatei definiert und wird mit der Endung .h gespeichert.

6.2 Editor „Properties“

/D20/ Die formale Eigenschaft, derer das Wahlverfahren genügen muss ist als C-Datei definiert, die einmal die Methode voting(params) aus einer Headerdatei aufruft, und wird mit der Endung .c gespeichert.

7 Nichtfunktionale Anforderungen

/F10/ Nicht mehr als 0,5 Sekunden Verzögerung bei Erfragen der Code-Comletion

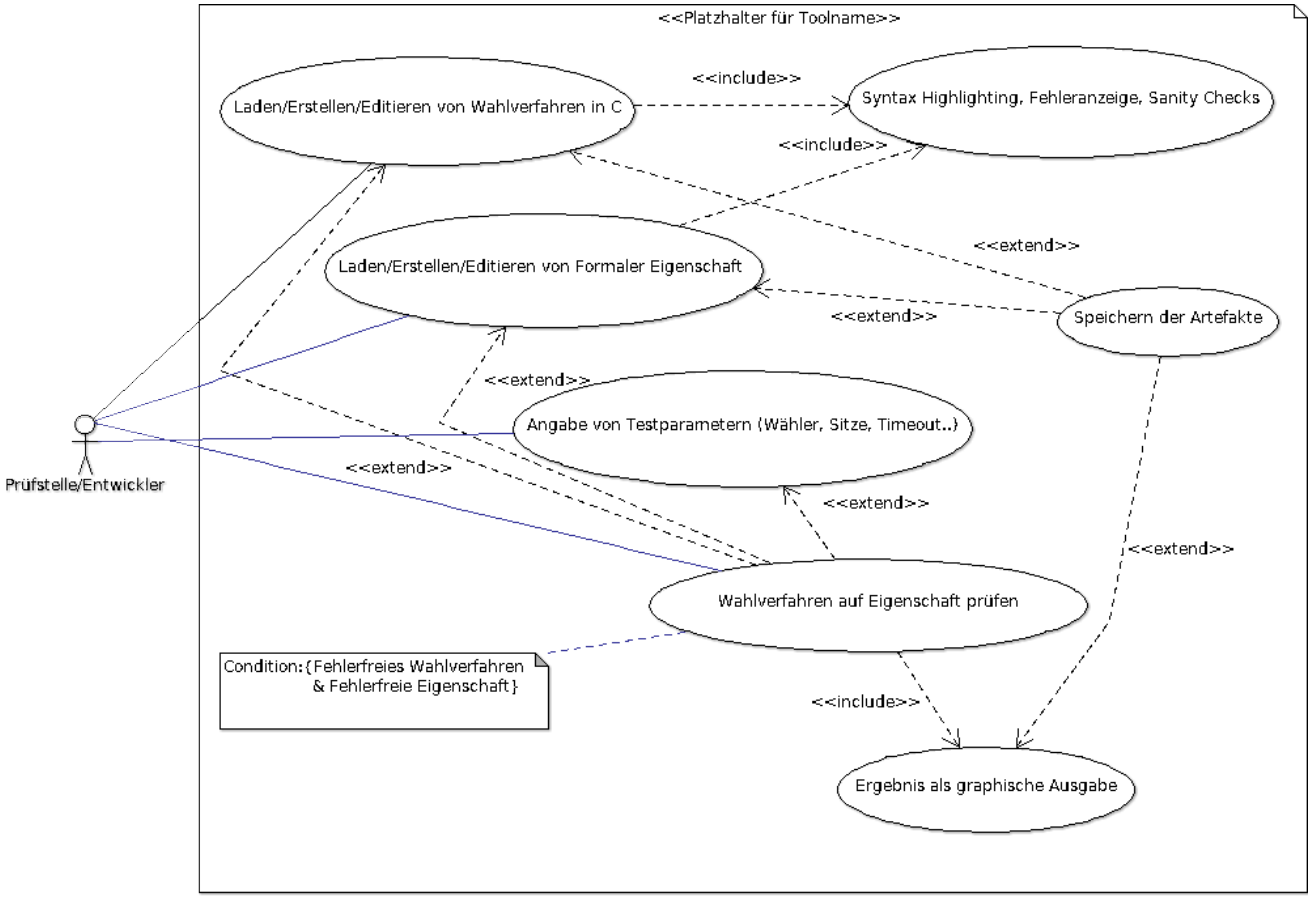
8 Globale Testfälle

9 Systemmodelle

9.1 Szenarien

9.2 Anwendungsfälle

Eine Prüfstelle oder ein Entwickler gibt dem Tool ein Wahlverfahren in C und eine formale Eigenschaft über beziehungsweise entwickelt diese selbst in den jeweiligen Editoren des Tools. Werden diese als plausibel erkannt kann er das Wahlverfahren, auch mit Angabe eigener Testparameter (Wähler, Sitze, Timeout...), auf die Eigenschaft prüfen und erhält das Ergebnis als graphische Ausgabe. Diese kann dann, sowie auch Wahlverfahren und formale Eigenschaft, abgespeichert werden.



9.3 Objektmodelle

9.4 Dynamische Modelle

10 GUI

11 Phasenverantwortliche

11.1 Pflichtenheft

Justin Hecht

11.2 Entwurf

11.3 Implementierung

11.4 Qualitätssicherung

11.5 Abschlusspräsentation

12 Glossar

Abbildungsverzeichnis