Karlsruher Institut für Technologie
Institut für Theoretische Informatik (ITI)
Prof. Dr. Bernhard Beckert
Sarah Grebing, Michael Kirsten

Praxis der Softwareentwicklung, WS 2016/17

Aufgabenbeschreibung: Analyse formaler Eigenschaften von Wahlverfahren

Eine kurze Bemerkung vorab

Dieses PSE-Projekt ist *Ihr* Projekt, in dem *Sie* eine Software-Anwendung konzipieren, entwerfen, umsetzen, testen und ausliefern werden.

Sie müssen und sollen selbst kreativ entscheiden, wie Sie *Ihr* Produkt gestalten. Dieses Dokument ist daher nicht als Katalog von Aufgaben zu verstehen, die Punkt für Punkt abgearbeitet werden müssen, um das Modul zu bestehen. Sie erhalten hier von uns Hinweise bezüglich unserer Vorstellungen und Erwartungen.

1 Hintergrund

Ein Wahlverfahren oder auch Wahl*auszähl*verfahren ist eine Methode, individuelle Präferenzen zu einer aggregierten Wahlentscheidung zu kombinieren. Häufig ist ein solches Verfahren Teil des grundsätzlichen demokratischen Verfahrens und stellt einen wichtigen Faktor für die Glaubwürdigkeit der gesamten Wahl dar. Es ist also unerlässlich, dass Wahlverfahren wie vorgesehen funktionieren, also beispielsweise den in einer rechtlich verbindlichen Verfassung spezifizierten Anforderungen genügen. Existierende Wahlverfahren haben ungewollte und manchmal überraschende Eigenschaften aufgewiesen (z. B. negative Stimmgewichte bei den Bundestagswahlen in Deutschland). Für kompliziertere, neu-designte Wahlverfahren, die mithilfe der Unterstützung von Computern im Stimmauszählverfahren ermöglicht werden, ist es wahrscheinlich, dass sich solche Sachverhalte ebenso abzeichnen.

2 Ziel

Ziel dieses Projekts ist es, ein graphisches Wahl-Analyse-Tool zu entwickeln, das es etwa der Entwicklerin eines Wahlverfahrens oder einer entsprechenden Prüfstelle erlaubt, ein Wahlverfahren komfortabel in der Sprache C zu erstellen und formale Eigenschaften hierfür zu schreiben, sowie beide Artefakte zu bearbeiten und zu analysieren. Ein Nichterfüllen der formalen Eigenschaft soll (in Form eines Gegenbeispiels) dargestellt werden.

3 Aufgabenstellung

Ihr Programm soll es erlauben, Wahlverfahren (als C-Programme) einzugeben und zu bearbeiten. Ferner soll es möglich sein, formale Eigenschaften über das jeweilige Wahlverfahren eingegeben und bearbeitet werden können. Diese Eigenschaften sollen als boolesche C-Ausdrücke in Form von Vor- und Nachbedingung ausgedrückt werden. Hierzu sollen spezielle Hilfs-Makros verwendet werden können. Weiterhin soll es in der Spezifizierung der formalen Eigenschaften erlaubt sein, Aussagen über symbolische Variablen formulieren zu können.

Hierbei ist es auch Ihre Aufgabe, diese Eigenschaften inklusive Wahlverfahren in eine sinnvolle Eingabe für CBMC zu übersetzen. Bei den Hilfs-Makros ist eine Beschränkung auf eine Quantifizierung über Wählern, Kandidaten bzw. Sitzen, einen Summenoperator über Wählern, Kandidaten bzw. Sitzen, sowie Implikations- und Äquivalenzoperatoren ausreichend. Bedenken Sie, dass CBMC zusätzlich zur C-Sprache nur mit nichtdeterministischen (symbolischen) Variablen sowie `assume`- und `assert`-Ausdrücken umgehen kann. Quantoren-Makros müssen in entsprechende C-Konstrukte mit Schleifen übersetzt werden.

Die Routine, die für ein Wahlverfahren überprüft, ob es eine formale Eigenschaft erfüllt, ist *nicht* Teil Ihrer Aufgabe, sondern sie wird Ihnen ihn Form des Werkzeugs `CBMC` zur Verfügung gestellt, das Sie ansteuern können.

Die Software soll möglichst robust gegen fehlerhafte und unerwartete Eingaben sein. So sollen z. B. Fehler im Entwurf des Wahlverfahrens sowie in der Spezifikation einer formalen Eigenschaft durch Ihr System abgefangen werden, damit die Benutzerin frühzeitig auf potentielle Probleme hingewiesen wird.

3.1 Minimale Leistungsmerkmale

1. **Ein Code-Editor für Wahlverfahren in der Programmiersprache *C*.**

   Laden/Speichern

   Syntax-Highlighting?

   Fehler-Anzeige?

2. **Ein Editor zur Spezifikation formaler Eigenschaften in abgespeckter C-Syntax mit speziellen Macros.** Abgespeckte C-Syntax mit Macros bedeutet nur C-Expressions vom Typ `boolean`, ohne Schleifen o. Ä., mithilfe eigens definierter Macros[1] und einer Eingabemaske für zusätzliche (neben Stimmen-Array(s) und Wahlergebnis(sen)) symbolische Variablen (Kandidaten, Wähler und Sitze).

   Laden/Speichern

   Syntax-Highlighting?

   Fehler-Anzeige?

   Code completion?

   Einfache Sanity checks

3. **Ansteuerung des Analysewerkzeugs CBMC.**

4. **Eine graphische Eingabemöglichkeit der zu analysierenden Anzahl von Wählern, Kandidaten und Sitzen.** Implementierung eines Timeouts? Optional: Angabe von **Intervallen** (ergo: Batch-Ansteuerung von CBMC).

5. **Eine graphische Darstellung von Gegenbeispielen.** Oder eine Ausgabe der erfolgreichen Analyse für die angegebenen Parameter.

6. Optional: **Eingabemöglichkeit konkreter Stimmen-Arrays mit Ausgabe des Wahlergebnisses[2] zu Testzwecken.**

[1] Hier sollen zumindest die Macros `FORALL_VOTERS[i]`, `FORALL_CANDIDATES[i]`, `FORALL_SEATS[i]` (jeweils mit einstellbarer quantifizierter Variable), `VOTE_SUM_FOR_CANDIDATE(c)` (c steht hier für eine symbolische Variable, die für einen Kandidaten steht), sowie logische Implikation (`==>`) und Äquivalenz (`<==>`) unterstützt werden. Eine einfache Definition weiterer Macros sollte im Code gegeben sein.

[2] Mittels eines einfachen „`assert(0);`“ direkt vor der Rückgabe des Ergebnisses können Sie das CBMC-Trace für die Rückgabe des Ergebnisses nutzen.

Software-Architektur. Für ein Projekt, in dem die Korrektheit der Analyse eine extrem hohe Bedeutung hat, ist eine saubere Trennung zwischen kritischen (Ansteuerung/Auswertung von CBMC) und unkritischen (z. B. GUI, aber auch Berechnung anderer Daten wie Statistiken) Komponenten von immenser Bedeutung. Entwerfen Sie austauschbare Komponenten mit minimalen, wohldefinierten Schnittstellen.

3.2 Zusatzmerkmale über die Mindestanforderungen hinaus

Es gibt eine Reihe von weiteren sinnvollen Merkmalen. Entscheiden Sie selbst und erörtern Sie im Pflichtenheft, welche davon für Sie in Frage kommen. Grundsätzlich gilt: Was im Pflichtenheft erwähnt wird, muss später auch umgesetzt werden.

Mögliche Zusatzmerkmale sind:

- Möglichkeit zur Ansteuerung weiterer SAT/SMT-Solver (standardmäßig verwendet CBMC den Solver `MiniSAT`)
- Sinnvolles Format zum Laden/Speichern der Eingabe inklusive Analyseergebnisse
- Aggregierte Anzeige verschiedener Laufzeiten für verschiedene Parameter
- Weitere Merkmale, die *Sie* sich für das Produkt als sinnvoll vorstellen

3.3 Technische Eckpunkte

- Programmiersprache: Java
- GUI-Bibliothek: Swing oder JavaFX
- Versionsverwaltung: Subversion (siehe unten) und eventuell GIT
- Entwurfs- und Entwicklungsumgebung: Ihre Entscheidung, z. B. ArgoUML
- Unit-Test-Rahmenwerk: JUnit
- Dokumenterstellung: Ihre Entscheidung, bevorzugt LaTeX (Abgabe aber immer als PDF)

4 Organisatorisches

4.1 Technische Ausstattung

Die Website zur Veranstaltung findet sich unter `http://formal.iti.kit.edu/teaching/pse/201617/voting/` Dort werden in Zukunft weitere organisatorische Informationen bereitgestellt.

In Raum 201 stehen Ihnen zur Zeit drei Rechnerarbeitsplätze zur Verfügung. Sie können aber selbstverständlich auch von zuhause (oder ATIS, SCC, Schlossgarten, ...) arbeiten.

Wir stellen ein SVN-Repository bereit, in dem alle Artefakte (insbesondere Abgaben) abgelegt werden sollen. Die Checkout-Adresse lautet `https://svnserver.informatik.kit.edu/i57/svn/pse16-2`. Wir haben für Sie Benutzeraccounts nach dem Schema `vorname_nachname` (ohne Zweitnamen) mit Ihrem Nachnamen in Kleinbuchstaben als Default-Passwort angelegt. Prüfen Sie bitte Ihren Zugang und ändern Sie möglichst bald Ihr Passwort. Für die Implementierung selbst können Sie auch auf Anbieter einer GIT-Versionsverwaltung (z. B. github.com, gitlab.com, bitbucket.org, ...) ausweichen, da sich hier insbesondere bei einem Programmier-Projekt im Team divergierende Versionen erfahrungsgemäß leichter zusammenbringen lassen. Hier ist Ihnen die Wahl des Anbieters freigestellt, in jedem Fall ist aber den Betreuern von Beginn an Zugriff zu gewähren.

4.2 Bewertung

Die Benotung Ihres Systems richtet sich nach folgenden Kriterien:[3]

- Qualität aller abgegebenen Dokumente
- Qualität der Kolloquien
- Qualität der Abschlusspräsentation
- Erfüllen der minimalen Leistungsmerkmale (s.o.)
- *Sinnvolle*[4] Erweiterungen über diese Merkmale hinaus
- Qualität des erstellten Programms (das schließt u.A. Benutzbarkeit und Robustheit ein)

Die Gesamtnote errechnet sich dann nach der im Modulhandbuch genannten Gewichtung:

- Pflichtenheft 10%
- Entwurf 30%
- Implementierung 30%
- Qualitätssicherung 20%
- Abschlusspräsentation 10%

Nach jeder Phase findet (im Rahmen der regelmäßigen Treffen) ein Kolloquium statt, in dem die Ergebnisse der Phase *selbstständig* (in rund 20 Minuten) vorgestellt werden sollen. Jedes Team-Mitglied muss einmal präsentieren.[5] Die Benotung jeder einzelnen Phase wird im entsprechenden Kolloquium besprochen.

4.3 Treffen

Treffen finden wöchentlich im Raum 201 statt. Auch wenn gerade kein Kolloquium ansteht, raten wir Ihnen dringend, jede Woche über Ihren Fortschritt zu berichten. Nur dann können wir unverbindliche, gezielte Kommentare abgeben. Sorgen Sie daher bitte auch dafür, dass alle erforderlichen Dokumente rechtzeitig und eindeutig identifizierbar im SVN verfügbar sind. Für den Abschluss jeder Phase gilt ein *strikter* Abgabeschluss um 11 Uhr *am Montag vor* dem jeweiligen Kolloquium.

Darüber hinaus sollten Sie sich natürlich auch noch regelmäßig in der Gruppe besprechen. Sie können dazu Raum 211 nutzen falls er frei ist (vor der Tür hängt ein Kalender mit Reservierungen; bitte nehmen Sie Rücksicht auf die Einträge und beachten Sie, dass Belegungen von Mitarbeiter*innen Priorität haben).

[3] Diese Liste hat keine Reihenfolge, die einer Gewichtung entspricht. Es gibt sicherlich weitere Punkte, die als selbstverständlich gelten und sich bei Nichterfüllen negativ auswirken.

[4] Sie tun sich selbst und dem Projekt nichts Gutes wenn Sie sich zu sehr verkünsteln.

[5] Dennoch müssen sich selbstverständlich *alle* Team-Mitglieder in *allen* Phasen einbringen.