Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. B. Beckert
Thorsten Bormer, Daniel Bruns

Praxis der Softwareentwicklung, WS 2012/13

Aufgabenbeschreibung

1 Eine kurze Bemerkung vorab

Dies ist *Ihr* Projekt. Dieses Dokument ist kein Katalog von Aufgaben, der Punkt für Punkt abgearbeitet werden muss, um das Modul zu bestehen, sondern lediglich eine Reihe von Hinweisen, was wir erwarten. Wie *Ihr* Programm nachher aussieht, müssen Sie selbst entscheiden.

2 Ziel

Ziel des Projekts ist es, ein mächtiges Analysewerkzeug für eine einfache imperative Programmiersprache zu erstellen. Dabei sollen sowohl klassische Techniken zur Programmanalyse zum Einsatz kommen (wie interaktives Debugging), als auch moderne, logikbasierte Methoden zur Analyse und Prüfung der Programmeigenschaften.

Um Methoden der Softwareverifikation implementieren zu können, die dem aktuellen Stand der Technik entsprechen, beschränken wir uns auf eine einfache Zielsprache, sowie auf eine einfache Spezifikationssprache, in der die gewünschten Eigenschaften des Programms angegeben werden. Die Steuerung der Komponenten des Analysewerkzeugs und dessen Rückmeldungen sollen dabei mit Hilfe einer grafischen Benutzeroberfläche erfolgen.

3 Aufgabenstellung

3.1 Minimale Leistungsmerkmale

Programmier- und Spezifikationssprache. Die Sprache wird von Ihnen selbst in Syntax und Semantik gewählt. Sie muss die Datentypen Ganzzahl und Boolean mit den üblichen Operationen sowie Kontrollstrukturen (`if` und `while`) umfassen. An Spezifikationen sollen zumindest Vor- und Nachbedingungen sowie Sicherheitsannotationen (security level) vorhanden sein.

Analysemethoden. Zur Analyse sollen zwei Methoden implementiert werden: 1. Interpretation mit Laufzeitprüfung von Spezifikationen und 2. (beschränkte) formale Verifikation durch Konstruktion der schwächsten Vorbedingung und automatischer Beweisführung in Z3. Für die Interpretation müssen Sie den Maschinenspeicher modellieren. Bei der *beschränkten* Verifikation wird das Programm zunächst soweit transformiert, dass Schleifen durch eine beschränkte Anzahl geschachtelter `if`-Anweisungen ersetzt werden. Dann findet die bekannte Methode der schwächsten Vorbedingung (wp) Anwendung, die in diesem Fall vollautomatisch abläuft. Z3 ist wiederum ein vollautomatischer Beweiser für Prädikatenlogik, der für falsche Aussagen auch Gegenbeispiele liefern kann.

Informationsflussanalyse. In einer Informationsflussanalyse wird geprüft, ob öffentliche Programmausgaben von geheimen Eingaben abhängen. Auf Sprachebene sind dabei Methodenparameter und -rückgabewerte oder global Variablen aus Ein- und Ausgabe zu verstehen. Ihre Spezifikationssprache muss die Annotation von Variablen, Parametern und Methoden mit Sicherheitsstufen erlauben. Dabei reicht es, zwei Sicherheitsstufen zu betrachten. Zur Beweisführung wird die Technik der Selbstkomposition benutzt, bei der ein Programm kopiert wird und in einer Kopie alle Variablen umbenannt werden. Danach kann mit der oben beschriebenen Methode der schwächsten Vorbedingung bewiesen werden, dass es keinen unzulässigen Fluss gibt.

Grafische Oberfläche. Ihr Programm muss über eine grafische Benutzeroberfläche verfügen. Diese enthält zumindest eine Editoransicht für ein Programm und die Steuerelemente, über die die oben genannten Analysemethoden aufgerufen werden.

Zusatzmerkmale über die Mindestanforderungen hinaus. Es gibt eine Reihe von weiteren Features, die als sinnvoll erachtet werden können. Entscheiden Sie selbst, bzw. erörtern Sie im Pflichtenheft, welche davon für Sie in Frage kommen. Grundsätzlich gilt: Was im Pflichtenheft erwähnt wird, muss später auch implementiert werden. Mögliche Bonusfeatures sind beispielsweise:

- Syntaxhervorhebung im Editor
- Visualisierung von Informationsflüssen im Editor
- Weitere Datentypen wie Arrays
- Globale Variablen (Felder)
- Deklassifikation von Geheimnissen
- Interaktives Debugging
- Speichern und Laden von Beweisen
- ...

3.2 Technische Eckpunkte

- Programmiersprache: Java
- GUI-Bibliothek: Swing
- Beweiser: Z3
- Versionsverwaltung: Subversion
- Entwurfs- und Entwicklungsumgebung: nach Wunsch
- Unit-Test-Rahmenwerk: JUnit
- Dokumenterstellung: nach Wunsch, bevorzugt LaTeX (Abgabe aber immer als PDF)

4 Organisatorisches

Die Website zur Veranstaltung findet sich unter `http://formal.iti.kit.edu/teaching/pse/ws201213/`. Dort werden in Zukunft weitere organisatorische Information bereitgestellt.

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

5 Bewertung

Die Benotung Ihres Systems richtet sich nach folgenden Kriterien:[1]

- Qualität aller abgegebenen Dokumente
- Qualität der Kolloquien
- Qualität der Abschlusspräsentation
- Erfüllen der minimalen Leistungsmerkmale (s.o.)
- *Sinnvolle*[2] 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 Kolloquim statt, in dem die Ergebnisse der Phase *selbstständig* (in rund 20 Minuten) vorgestellt werden sollen. Jedes Team-Mitglied muss einmal präsentieren.[3] Die Benotung jeder einzelnen Phase wird im entsprechenden Kolloquium besprochen.

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

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

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