Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. Bernhard Beckert
Thorsten Bormer, Dr. Vladimir Klebanov, Dr. Mattias Ulbrich

Formale Systeme, WS 2014/2015

Praxisaufgabe 2: Theorembeweiser

Abgabe der Lösungen bis zum 12.02.2015 über die ILIAS-Seite zur Vorlesung
`https://ilias.studium.kit.edu/goto_produktiv_crs_374019.html`

Hinweis: Bitte beachten Sie, dass die Praxisaufgabe in Einzelarbeit und selbständig zu bearbeiten ist. Wir behalten uns vor, die selbständige Bearbeitung dadurch stichprobenartig zu prüfen, dass wir uns die eingereichte Lösung erklären lassen.

Für die vollständige Lösung dieser Praxisaufgabe erhalten Sie **20 Übungspunkte**. Bitte beachten Sie die Erläuterung zu Übungspunkten auf der Webseite zur Vorlesung. Für unvollständige Lösungen werden die Übungspunkte anteilig vergeben.

A Informationen zum KeY-System

A.1 Allgemeines

Was ist KeY? Zusammen mit unseren Partnern, unter anderem an der Technische Universität Darmstadt, wird an unserem Institut das KeY-System entwickelt. Es ist ein Softwareverifikationswerkzeug, mit dem die Übereinstimmung von Java Card-Software und ihrer Spezifikation formal bewiesen werden kann.

Die Logik, auf der das KeY-System basiert, ist eine sortierte dynamische Prädikatenlogik. In dieser dynamischen Logik ist die in der Vorlesung eingeführte Prädikatenlogik vollständig enthalten. Deshalb können wir KeY auch benutzen, um rein prädikatenlogische Probleme zu formulieren und zu beweisen.

Literatur zu KeY Auf der Internetseite der Vorlesung steht eine Einführung zu Beweisen von prädikatenlogischen Formeln mit KeY (`KeYIntro.pdf`). Bitte arbeiten Sie diese Einführung durch.

Für weitergehende Fragen bietet sich die Lektüre des KeY-Buches [BHS07] an und darin vor allem Kapitel 10, das eine tiefere Einführung in KeY bietet. Kapitel 2 des Buches erklärt fundiert die prädikatenlogischen Grundlagen, auf denen KeY basiert.

Installation Das KeY-System besitzt eine graphische Benutzeroberfläche und ist komplett in Java geschrieben, so dass es auf jeder Plattform, für die eine virtuelle Maschine für Java zur Verfügung steht, lauffähig ist.

Auf der Webseite zu dieser Vorlesung können Sie die Version von KeY finden, die Sie zur Lösung dieses Übungsblattes verwenden sollen. Wenn Sie die Software „Java Web Start“ installiert haben (auf fast allen modernen Systemen mit Java der Fall), können Sie KeY direkt aus dem Internetbrowser heraus starten.

Bei neueren Java-Versionen sind die Sicherheitseinschränkungen verschärft worden. Es kann daher sein, dass Sie das Programm nicht über Web Start starten können. Wir stellen auf der Vorlesungswebseite daher auch eine herunterladbare Version von KeY zur Verfügung.

Bitte verwenden Sie in jedem Fall die KeY-Version von der Vorlesungshomepage und nicht die Versionen auf den Seiten des Tools.

Hinweis zur Konfiguration von KeY Für die Aufgaben dieses Blattes empfiehlt es sich, die Standardeinstellungen des KeY-Systemes zu belassen, wie sie zum Zeitpunkt des Systemstarts sind.

A.2 Abgabe der Lösungen

Bitte speichern Sie alle Beweise in KeY als Dateien mit der Namenserweiterung `.key.proof` ab. Auf der ILIAS-Seite der Vorlesung stehen Ihnen Formulare zur Verfügung, um die Lösungsdateien einzureichen.

B Teilaufgabe 1: Teilbar durch 3 — 7 Punkte

Die erste Aufgabe wird sich mit einem einfachen Satz aus der Arithmetik beschäftigen, den es mit Hilfe von KeY maschinell zu beweisen gilt. Es wird kein Java-Programm in dieser Teilaufgabe vorkommen, dennoch kann der Beweis nicht vollautomatisch von KeY gefunden werden, und es müssen einige Beweisschritte interaktiv durchgeführt werden.

Der einfache mathematische Satz, den Sie in dieser Aufgabe beweisen sollen, lautet:

Satz 1 *Für jede natürliche Zahl $n \in \mathbb{N}$ ist der Ausdruck $n^3 - n$ durch 3 teilbar.*

B.1 Formalisierung und Eingabedatei

Der Theorembeweiser KeY hat eingebaute Unterstützung für die *ganzen* Zahlen $\mathbb{Z}$ über den Datentyp `int`. Formulieren Sie damit die Aussage aus Satz 1 in Prädikatenlogik erster Stufe über den ganzen Zahlen.

Erstellen Sie eine KeY-Eingabedatei `threeDivides.key`, die die Aussage als Beweisverpflichtung enthält. Die Definition der Syntax von KeY-Dateien finden sie in der oben erwähnten Einführung für KeY. Beachten Sie, dass es *keinen* Operator für die Exponentation (x^m) in KeY gibt.

B.2 Beweis und Beweisdatei

Führen Sie nun den Beweis für Satz 1 in KeY und speichern Sie den Beweis in einer Datei mit dem Namen `threeDivides.key.proof` ab.

Die Behandlung der Arithmetik in KeY ist sehr mächtig und kann viele Aussagen vollautomatisch beweisen. Die Menge der eingebauten arithmetischen Regeln ist groß und geht aus Effizienzgründen weit über die notwendigen Axiome (vgl. Axiome der *Peano-Arithmetik* in Kapitel 5.8 im Skript). Daher ist es bisweilen als menschlicher Benutzer schwierig, den automatisch angewandten Beweisschritten zu folgen. Es empfiehlt sich also, insbesondere zu Beginn eines Beweises viele Schritte manuell durchzuführen. Während des weiteren Beweises ist es dann sinnvoll, zu überprüfen, ob das Problem schon soweit vereinfacht wurde, dass die Automatik es erfolgreich lösen kann. Wählen Sie dafür den Punkt „Close provable goals below“ aus dem Kontextmenü „Strategy Macros“, das in der Sequenzen- oder der Beweisbaumansicht verfügbar ist[1].

Das 7. Axiom der in der Vorlesung vorgestellten *Peano-Arithmetik* (Def. 5.88 im Skript) ist zentral und ermöglicht das Führen von Beweisen mittels „vollständiger Induktion“. Die folgende Regel setzt das Erste-Stufe-Induktionsprinzip des Axioms im Sequenzenkalkül um. Da der eingebaute Typ `int` die ganzen Zahlen und nicht die natürlichen Zahlen modelliert, sind die Einschränkungen $z \geq 0$ für die Korrektheit notwendig.

$$\frac{\Gamma \Rightarrow \varphi[z/0], \Delta \qquad \Gamma \Rightarrow \forall z\ z \geq 0 \rightarrow (\varphi \rightarrow \varphi[z/z+1]), \Delta \qquad \Gamma, \forall z\ z \geq 0 \rightarrow \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

Diese Regel trägt im KeY-System den Namen „int_induction“ und kann angewendet werden, indem man in der Sequenzenanzeige auf den Sequenzenpfeil klickt. In das Eingabefenster, dass sich dann öffnet, muss die Induktionshypothese φ eingetragen werden und der Name der Induktionsvariablen kann gewählt werden (Standard ist *nv*).

[1] Sie können auch die rechte Maustaste in der Sequenzenansicht verwenden.

B.3 Abgabe

Bitte laden Sie die Datei `threeDivides.key.proof` mit dem Beweis der Aussage in das dafür vorgesehene Formularfelder in ILIAS hoch[2].

B.4 Verallgemeinerung *(keine Abgabe – keine Bewertung)*

Beweisen Sie mit KeY die folgende Verallgemeinerung, indem Sie Satz 1 als Lemma verwenden:

Satz 2 *Für jede* ganze *Zahl* $z \in \mathbb{Z}$ *ist der Ausdruck* $z^3 - z$ *durch drei teilbar.*

C Teilaufgabe 2: Länge des Null-Präfix in einem Array bestimmen 13 Punkte

In dieser Aufgabe soll das gewünschte Verhalten eines kleinen Java-Programms in JML formalisiert werden. Und es soll mit KeY bewiesen werden, dass das Programm sich tatsächlich so verhält. Um die Aufgabe zu formulieren, führen wir zunächst den Begriff des Null-Präfixes ein.

Gegeben sei ein `int`-Array. Das *Null-Präfix* des Arrays ist das längste zusammenhängende Anfangsteilstück des Arrays, das nur Nullen enthält. Beispiel: Das Null-Präfix des Arrays `[0,0,2,0,5,0,0,0]` ist `[0,0]`.

Aufgabe

(a) Auf der Vorlesungswebseite finden Sie das folgende Klassenskelett:

```
class ZeroPrefix {

    public static int zeroPrefixLength(int[] a) {
    //...
    }

}
```

Implementieren Sie die Methode `zeroPrefixLength(int[] a)` so, daß sie die Länge des Null-Präfix des Eingabe-Arrays `a` zurückgibt.

(b) Spezifizieren Sie das gewünschte Verhalten der Methode in JML (Java Modeling Language).

(c) Beweisen Sie die Korrektheit der Implementierung gegenüber der JML-Spezifikation. Fügen Sie dafür ggf. zusätzliche JML-Annotationen (wie bspw. Schleifeninvarianten) hinzu.

(d) Speichern Sie den geschlossenen Beweis in einer Datei mit dem Namen `zeroprefix.key.proof` ab.

Abgabe Bitte laden Sie die vervollständigte Datei `ZeroPrefix.java` mit annotiertem Programm, sowie die Datei `zeroprefix.key.proof` mit dem Beweis in die dafür vorgesehenen Formularfelder in ILIAS hoch.

[2]Diese Datei enthält neben dem Beweis auch Ihre ursprüngliche Formalisierung, so dass Sie die Datei `threeDivides.key` nicht ebenfalls einreichen müssen.

Weitere Hinweise Sollte Ihre Implementierung und Spezifikation konsistent sein, und die notwendigen Zusatzannotationen hinreichend, so funktioniert der Beweis in der Regel automatisch (z.B. durch Betätigen des grünen „Play“-Knopfes).

Erfahrungsgemäß klappt Beweisen aber nicht auf Anhieb, da die Implementierung, die Spezifikation, und die Zusatzannotationen Fehler enthalten können. Zum „Debuggen“ empfiehlt es sich, den „Beweis-Autopiloten“ zu nutzen. Wählen Sie dafür nach Laden der Beweisverpflichtung den Punkt „Full Auto Pilot“ aus dem Kontextmenü „Strategy Macros“, das in der Sequenzen- oder der Beweisbaumansicht verfügbar ist. Der Autopilot strukturiert den Beweis stärker anhand der zu beweisenden Aussage: jeder Einzelteil der zu beweisenden Aussage wird als separates Beweisziel behandelt. An den offenen Zielen kann man in diesem Fall besser erkennen, welcher Teil der Korrektheitsaussage nicht (automatisch) bewiesen werden konnte.

KeY wird mit einigen Beispielen ausgeliefert, diese können unter „Load Example“ im File-Menü aufgerufen werden. Als Einstieg sind die Beispiele „Sum and Max“, „Binary Search“, und „Remove Duplicates“ gut geeignet.

Literatur

[BHS07] Bernhard Beckert, Reiner Hähnle, and Peter H. Schmitt, editors. *Verification of Object-Oriented Software: The KeY Approach.* LNCS 4334. Springer-Verlag, 2007.