Universität Koblenz-Landau WS '03/04
Institut für Informatik
Bernhard Beckert · `www.uni-koblenz.de/~beckert`
Vladimir Klebanov · `www.uni-koblenz.de/~vladimir`

Praktikum

Entwicklung objektorientierter Software mit formalen Methoden

Aufgabenblatt 7

Aufgabe 15

Laden Sie sich das Archiv mit den Problemdateien von der Praktikums-Webseite herunter. Legen Sie ein Verzeichnis für die Aufgaben an, legen Sie die Datei dort hinein, und entpacken Sie sie mit dem Befehl:

```
tar -xzvf aufg15.tgz
```

Sie erhalten 4 Dateien mit Namen `p15_1.key` bis `p15_4.key`. Wie schon in Aufgabe 13 enthält jede dieser Dateien eine Beweisaufgabe, die Sie mit dem KeY-Beweiser lösen sollen. Verfahren Sie zum Laden und Beweisen der Probleme wie in Aufgabe 13 beschrieben. Zur Abgabe gehen Sie ebenfalls wie in Aufgabe 13 vor.

`p15_1.key` Hier soll gezeigt werden daß der Wert einer `int` Variablen `i` in einem Java Programm nach Ausführung der Anweisung `i++` um eins höher ist als vorher. Schalten Sie, bevor Sie anfangen zu beweisen, die Heuristik `simplify_updates` ein, indem sie diese mit dem Menüpunkt `Proof | Heuristics` aktivieren.

Ansonsten geht es hier darum, schrittweise die Programmformeln abzuarbeiten. Befindet sich der Mauszeiger über einer Programmformel, wird die jeweils nächste zu behandelnde Anweisung grau unterlegt. Meist ist nur eine Regel anwendbar, etwa `eliminate_variable_declaration`, `assignment`, etc.

`p15_2.key` Hier sollen Sie zeigen, dass nach Ausführung des Programmstücks mit den lokalen Variablen `i` und `k`:

```
if (k==0) {
  i=k;
}
else {
  i=0;
}
```

`i` den Wert 0 hat. Machen Sie wieder alle Schritte ohne Hilfe der Heuristiken (mit Ausnahme von `simplify_updates`). Sie werden Fallunterscheidungen machen, die Sie beim "intutitiven Überprüfen" des Programms auch machen würden (z.B. ob `k` den Wert 0 hat oder nicht). Denken Sie daran, dass eine Sequenz und eine Implikation auch dann wahr werden, wenn Sie deren linke Seite zu $false$ vereinfachen können.

`p15_3.key` Bevor Sie die Problemdatei in den KeY-Beweiser laden, öffnen Sie die Datei in einen Editor Ihrer Wahl (z.B. xemacs) und betrachten Sie sich die DL-Formel, die innerhalb von `problem{` ... `}` steht. Sie enthält folgendes Java-Programm in einem Diamond:

```
int i=0;
try {
   throw e;
   i=i+1;
} catch (RuntimeException e1) {
   i=i+4;
} finally {
   i=i+8;
}
```

Hierbei ist `e` ein Objekt (nicht mit `null` belegt) der Klasse `IllegalArgumentException`. `IllegalArgumentException` ist eine Unterklasse von `RuntimeException`. Ersetzen Sie in der hinter dem Diamond stehenden Teilformel `i = XXX` das `XXX` durch eine solche Zahl, dass die DL-Formel allgemeingültig ist. Wenn Ihnen die Java-Semantik bezüglich der Behandlung von Exceptions nicht mehr geläufig ist, so schlagen Sie in der Java Language Specification[1] nach.

Laden Sie dann die so veränderte Problemdatei in den KeY-Beweiser und weisen Sie die Allgemeingültigkeit nach.

(a) Machen Sie zu Beginn jede Regelanwendung ohne Zuhilfename des *Apply Heuristics*-Knopfes.

(b) Führen Sie noch die Regel `try_throw` aus und beschreiben Sie, was diese Regel gemacht hat.

(c) Führen Sie den Rest des Beweises, indem Sie alle Heuristiken einschalten und auf den Knopf *Apply Heuristics* klicken. Der Beweis läuft dann automatisch ab. Im linken Teil des Beweiserfensters können Sie verfolgen, welche Regeln von den Heuristiken angewandt wurden.

`p15_4.key` Zu dieser Aufgabe gehört die Java-Klasse `MyClass.java`, die Sie im Unterverzeichnis `classes` finden. Laden Sie erneut die .key-Datei in einen Editor und überlegen Sie sich, durch was Sie `XXX` ersetzen müssen, damit die Eingabeformel allgemeingültig ist. Natürlich sollte nicht schon die Teilformel hinter dem Diamond allein eine Tautologie sein. Beweisen Sie die Allgemeingültigkeit der veränderten Formel mit KeY.

Aufgabe 16 (`while` Regel)

Die folgende Regel (`unwind_while`)

$$\frac{\Gamma \vdash < \texttt{if (b) } \{\texttt{l1} : \{\texttt{l2} : \{\alpha'\} \texttt{ while (b)}\{\alpha\}\}\} > \Phi,\ \Delta}{\Gamma \vdash < \texttt{while (b)}\{\alpha\} > \Phi,\ \Delta}$$

führt genau einen Schleifendurchlauf aus. Dabei stimmt α' mit α überein, bis auf dass jedes in α vorkommende `break;` durch ein `break l1;` und jedes `continue;` durch ein `break l2;` ersetzt.

- Laden Sie das Problem `p16.key` in Ihren Editor. Im Abschnitt `problem` finden Sie ein Java Program. Überlegen Sie sich den Wert der Variablen i nach Ausführung des Programms und tragen Sie ihn anstelle der `XXX` ein. Laden Sie das Problem in den Beweiser, beweisen Sie Ihre Vermutung und speichern Sie die Lösung ab. Stellen Sie sicher, dass Sie die folgenden (und nur die) Heuristiken aktiviert haben: `simplify_updates, alpha_concrete, boolean_cases,`

[1] `http://java.sun.com/docs/books/jls/second_edition/html/jTOC.doc.html`

`simplify_int, simplify_literals`. Stellen Sie die Anzahl der maximalen Heuristikanwendungen auf 50 und vergewissern Sie sich, dass in der Werkzeugleiste 'Autoresume Heuristics' aktiviert (Häckchen) ist.

– Sehen Sie sich den Beweis aus der vorhergehenden Teilaufgabe in Ruhe an. Erklären Sie die Funktionsweise der `unwind_while` Regel.

Abgabe bis 26.1.2004

Es muss pro Gruppe nur *eine* Lösung abgegeben werden.
Die Abgabe der Übungsblätter erfolgt mit dem CVS System. Dazu legen Sie die abzugebenden Dateien im CVS ab und markieren die abzugebende Version der Dateien mit "`LoesungsBlatt`<*nr*>" wie in Aufgabe 2 beschrieben. Die Lösungen sollten vorzugsweise im dazugehörigen Unterverzeichnis `uebeungsblaetter/nr/` vorzufinden sein, zumindest aber ein Hinweis auf den Ort der Lösungen.
Einige Aufgaben verlangen eine schriftliche Bearbeitung, diese ist dann je nach Komplexität als ASCII, html, ps- oder pdf-Dokument abzugeben. Auf *keinen* Fall im MS Word `doc`-Format.

Materialien

`http://www.uni-koblenz.de/~beckert/Lehre/Praktikum-Formale-Entwicklung/`

Bernhard Beckert: Zi. MB 215, Tel. 287-2775, Email: `beckert@uni-koblenz.de`
Vladimir Klebanov: Zi. MB 224, Tel. 287-2781, Email: `vladimir@uni-koblenz.de`