Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. Peter H. Schmitt
Dr. V. Klebanov, Dr. M. Ulbrich, C. Scheben

Formale Systeme, WS 2013/2014

Praxisaufgabe 2: Theorembeweiser

Abgabe der Lösungen bis zum 05.02.2014 über die Webseite zur Vorlesung
`http://i12www.ira.uka.de/~pschmitt/FormSys/FormSys1314/`

Für die vollständige Lösung dieser Praxisaufgabe – bewertet wird in diesem Jahr nur Teilaufgabe 2 – erhalten Sie 1,5 Bonuspunkte für die Abschlussklausur. Bitte beachten Sie die Erläuterung zu Bonuspunkten auf der Webseite zur Vorlesung.

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, ohne etwas Weiteres installieren zu müssen.

A.2 Hinweis zur Konfiguration von KeY

Für die Aufgaben dieses Blattes empfiehlt es sich, die Standardeinstellungen des KeY-Systemes zu belassen, wie sie am Systemstart sind.

A.3 KeY-Problemdateien zu den Teilaufgaben

- Für Teilaufgabe 1 erstellen Sie bitte selbst eine KeY-Problemdatei.
- Für Teilaufgabe 2 finden Sie auf der Webseite der Vorlesung KeY-Problemdateien zum Herunterladen vor.

A.4 Abgabe der Lösungen

Bitte speichern Sie alle Beweise in KeY als „.key.proof“-Dateien ab. Auf der Webseite der Vorlesung steht Ihnen ein Formular zur Verfügung, um die Lösungsdateien einzureichen.

Für unvollständige Beweise werden Punkte auch anteilig vergeben.

B Teilaufgabe 1: Der Fall „Tante Agathe“ 0 Punkte

Folgendes logische Puzzle ist Teil einer großen Benchmarkbibliothek für Beweiser. Die natürlichsprachige Formulierung lautet:

> Tante Agathe wurde in ihrem Haus tot aufgefunden. Nach bisherigen Ermittlungen gilt Folgendes als sicher:
>
> 1. Im Haus lebten nur Agathe, ihr Butler und Onkel Charles.
> 2. Agathe wurde von einem Hausbewohner getötet.
> 3. Wer jemanden tötet, hasst sein Opfer.
> 4. Charles hasst niemanden, den Agathe hasste.
> 5. Der Täter ist niemals reicher als das Opfer.
> 6. Der Butler hasst alle, die nicht reicher als Agathe sind, sowie alle, die Agathe hasste.
> 7. Kein Bewohner des Hauses hasst(e) alle Hausbewohner.
> 8. Agathe hasste alle außer dem Butler.
> 9. Agathe war nicht der Butler.

Zur Formalisierung dieser Aussagen wählen wir die Signatur: $\Sigma_{Agathe} = (F, P, \alpha)$ mit

- $P = \{\text{kills}, \text{hates}, \text{richer}\}$
- $F = \{a, b, c\}$
- $\alpha(a) = \alpha(b) = \alpha(c) = 0, \qquad \alpha(\text{kills}) = \alpha(\text{hates}) = \alpha(\text{richer}) = 2$

Über diese Signatur lassen sich die gegebenen Indizien folgendermaßen formalisieren:

$$
\begin{aligned}
& \forall x(x \doteq a \vee x \doteq b \vee x \doteq c) \\
\wedge\ & \exists x(kills(x, a)) \\
\wedge\ & \forall x\ \forall y\ (kills(x, y) \rightarrow hates(x, y)) \\
\wedge\ & \forall x\ (hates(a, x) \rightarrow \neg hates(c, x)) \\
\wedge\ & \forall x\ \forall y\ (kills(x, y) \rightarrow \neg richer(x, y)) \\
\wedge\ & \forall x\ ((\neg richer(x, a) \vee hates(a, x)) \rightarrow hates(b, x)) \\
\wedge\ & \forall x\ \exists y\ \neg hates(x, y) \\
\wedge\ & \forall x\ (\neg x \doteq b \rightarrow hates(a, x)) \\
\wedge\ & \neg a \doteq b
\end{aligned}
$$

- Formalisieren Sie nun auch noch die Aussage „Tante Agathe hat sich selbst umgebracht.“,
- schreiben Sie eine Problembeschreibungsdatei für den Beweiser KeY und
- beweisen Sie mit KeY, dass aus den Indizien folgt, dass Tante Agathe sich selbst umgebracht haben muss.

Hinweis: KeY kann diesen Fall alleine lösen. Eine Musterlösung der Aufgabe wird mit KeY mitgeliefert: sie ist im Verzeichnis `examples/firstTouch/01-Agatha/` zu finden.

C Teilaufgabe 2: Abstrakte Datentypen 1,5 Punkt

In der Vorlesung wurde die *Peano-Arithmetik* vorgestellt, eine Formalisierung der natürlichen Zahlen in PL1. Sie finden sie im Skript in Kapitel 5.8.1.

Aufgabe. In dieser Aufgabe wollen wir aus den Peano-Axiomen sowie der Definition der Fibonacci-Zahlen ableiten, dass für alle natürlichen Zahlen n, m die $(n+m+1)$-te Fibonacci-Zahl der Summe aus dem Produkt der n-ten und m-ten Fibonacci-Zahl und dem Produkt der $(n+1)$-ten und $(m+1)$-ten Fibonacci-Zahl entspricht.

Signatur. Die Prädikatenlogik in KeY ist eine sortierte Logik. Wir betrachten im Folgenden nur die Sorte *nat* mit folgender Signatur:

Funktionen

$zero$	:	$\to nat$
one	:	$\to nat$
$plus$	:	$nat \times nat \to nat$
$times$	:	$nat \times nat \to nat$
fib	:	$nat \to nat$

Axiome. Die Funktionen *plus*, *times* und *fib* sind durch folgende Axiome (partiell) festgelegt:

Peano-Axiome (PA)

$\forall\ nat\ x$		$\neg plus(x, one) \doteq zero$
$\forall\ nat\ x$	$\forall\ nat\ y$	$plus(x, one) \doteq plus(y, one) \to x \doteq y$
$\forall\ nat\ x$		$plus(x, zero) \doteq x$
$\forall\ nat\ x$	$\forall\ nat\ y$	$plus(x, plus(y, one)) \doteq plus(plus(x, y), one)$
$\forall\ nat\ x$		$times(x, zero) \doteq zero$
$\forall\ nat\ x$	$\forall\ nat\ y$	$times(x, plus(y, one)) \doteq plus(times(x, y), x)$

Fibonacci-Axiome

	$fib(zero) \doteq zero$
	$fib(one) \doteq one$
$\forall\ nat\ n$	$fib(plus(n, plus(one, one))) \doteq (plus(fib(n), fib(plus(n, one))))$

Induktionsregel. Anstatt des Axiomenschemas für die Induktion verwenden wir folgende äquivalente Sequenzenkalkülregel:

$$\dfrac{\begin{array}{lll} \Gamma \Rightarrow \{n/zero\}\varphi,\ \Delta & & (1) \\ \Gamma \Rightarrow \forall\ nat\ n\ (\varphi \to \{n/plus(n, one)\}\varphi),\ \Delta & & (2) \end{array}}{\Gamma \Rightarrow \forall\ nat\ n\ \varphi,\ \Delta \quad (3)}\ \text{(induction\_on\_naturals)}$$

Um eine Aussage (3) über die natürlichen Zahlen mittels vollständiger Induktion zu beweisen, muss der Induktionsanfang (1) (engl. „*Base Case*“) und der Induktionsschritt (2) (engl. „*Step Case*“) gezeigt werden.

C.1 Wichtige Lemmata

Es ist schwer (wenn nicht gar unmöglich), die gewünschte Aussage mit Hilfe einer einzigen Anwendung von vollständiger Induktion zu bewerkstelligen. Deshalb haben wir den Beweis in mehrere Zwischenziele aufgeteilt. Ein bewiesenes Zwischenziel wird dann in der nachfolgenden Beweisaufgabe als Lemma zu den Annahmen mit hinzugenommen:

Lemma 1:	`addassoc.key`	*PA*	$\Longrightarrow$ *plus* ist assoziativ
Lemma 2:	`addcomm.key`	*PA, Lemma 1*	$\Longrightarrow$ *plus* ist kommutativ
Lemma 3:	`multcomm.key`	*PA, Lemma 1 - 2*	$\Longrightarrow$ *times* ist kommutativ
Lemma 4:	`dist.key`	*PA, Lemma 1 - 3*	$\Longrightarrow$ *plus* und *times* sind distributiv
Lemma 5:	`eq.key`	*PA, Lemma 1 - 4*	$\Longrightarrow plus(a,b) \doteq plus(a,c)$ g. d. w. $b \doteq c$

C.2 Die zu beweisende Aussage

Beweisen Sie zunächst mit KeY, dass die Lemmata korrekt sind. Hierzu finden Sie auf der Seite zur Vorlesung die KeY-Dateien `addassoc.key` (Lemma 1), `addcomm.key` (Lemma 2), `multcomm.key` (Lemma 3), `dist.key` (Lemma 4) und `eq.key` (Lemma 5).

Beweisen Sie anschließend mit KeY, dass für alle natürlichen Zahlen n, m die $(n+m+1)$-te Fibonacci-Zahl der Summe aus dem Produkt der n-ten und m-ten Fibonacci-Zahl und dem Produkt der $(n+1)$-ten und $(m+1)$-ten Fibonacci-Zahl entspricht:

$$\begin{aligned} &\forall\ nat\ n \quad \forall\ nat\ m \qquad fib(plus(n, plus(m, one))) \\ &\qquad\qquad\qquad\qquad \doteq plus(times(fib(n), fib(m)), times(fib(plus(n, one)), fib(plus(m, one)))) \end{aligned}$$

Auf der Seite zur Vorlesung finden Sie hierzu die KeY-Datei `fib.key`, in der die Peano-Axiome, die Fibonacci-Axiome, die obigen Lemmata 1 - 5, die Induktionsregel sowie das Beweisziel formalisiert sind:

`fib.key` *PA, FA, Lemma 1 - 5* $\Longrightarrow$ *Aussage*

Hinweis: Wir werden Ihnen später zeigen, wie obige Aussage mit KeY mit nur einer einzigen Interaktion bewiesen werden kann.

C.3 Eigene Lemmata

Einige der Beweisaufgaben kann das KeY-System automatisch lösen, wenn die Induktionsregel angewendet worden ist, andere benötigen einige interaktive Regelanwendungen.

Es kann nützlich sein, während des Beweises, lokal weitere Lemmata einzuführen. Benutzen Sie die (eingebaute) Regel `cut`, um ein Lemma einzufügen:

$$\frac{\overbrace{\Gamma, \varphi \Rightarrow \Delta}^{(A)} \qquad \overbrace{\Gamma \Rightarrow \varphi, \Delta}^{(B)}}{\Gamma \Rightarrow \Delta}\ \mathsf{(cut)}$$

`cut` teilt für eine Formel φ den Beweis in zwei Äste auf:

- Ast A: Hier steht die Formel φ *links* des Sequenzenpfeiles. Damit steht sie auf diesem Ast als weitere Voraussetzung zur Verfügung und kann benutzt werden, um das ursprüngliche Ziel zu schließen (Verwenden des Lemmas).
- Ast B: Hier steht φ *rechts* des Sequenzenpfeiles. Auf diesem Ast ist φ also das Beweisziel und muss gezeigt werden (Beweis des Lemmas).

Literatur

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