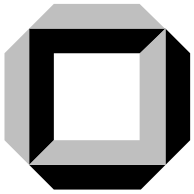

Universität Karlsruhe
Institut für Theoretische Informatik
Prof. Dr. P. H. Schmitt
C. Engel, B. Weiß

3. Dezember 2008

Praktikum
Formale Entwicklung objektorientierter Software
Übungsblatt 6

Aufgabe 12

Geben Sie analog zu den Folien die linke und die rechte Regel für die Äquivalenz $\leftrightarrow$ an.

Aufgabe 13

Laden Sie sich das Archiv `keyex1.tgz` von der Praktikums-Webseite herunter. Entpacken Sie die Datei mit dem Befehl:

```
tar xzvf keyex1.tgz
```

Sie erhalten im Verzeichnis `keyex1/` sieben Dateien mit Namen `p1.key` bis `p7.key`. Jede dieser Dateien enthält eine Beweisaufgabe, die Sie mit dem KeY-Beweiser lösen sollen.

Starten Sie den KeY-Beweiser. Auf Ihrem Praktikumsaccount benutzen Sie hierzu den Befehl `runProver`. Falls Sie von zuhause aus arbeiten, erhalten Sie die passende KeY-Version (1.4 TP) von folgender Webseite:

```
http://www.key-project.org/download/tp14.html
```

Sie können dort KeY entweder mittels Java Webstart über das Netzwerk starten, oder eine Bytecode-Version zur lokalen Installation herunterladen. Nach der Installation der Bytecode-Version kann KeY durch Aufrufen von `bin/startProver` gestartet werden.

Bitte stellen Sie sicher, dass im Menü unter Options — Minimize Interaction ein Häkchen gesetzt ist.

Problemdateien (Dateiendung `.key`) können jetzt mit dem Menüpunkt `File | Load` geladen werden. **Zur Abgabe** speichern Sie die Aufgaben mit `File | Save` als `p1.proof` bis `p7.proof`. Die Probleme sollen zunächst *ohne* Anwendung von "Strategien" gelöst werden, d.h. Sie sollen alle notwendigen Regeln *von Hand* anwenden. Versuchen Sie sich bei jedem Schritt darüber klar zu werden, was Sie gerade machen! Sie sollten auch in der Lage sein, Ihre Lösung vorzuführen. Es folgen einige Hinweise zu den einzelnen Problemen:

`p1.key` Zu beweisen ist die triviale Aussage `true`. Klicken Sie auf die Formel `true`. Es erscheint ein Menü mit möglichen Regelanwendungen. `closeTrue` schließt den Ast. Es erscheint ein Dialog mit der Meldung „Proved", d.h. der Beweis ist fertig. Das geschlossene Ziel ist in der linken Hälfte des Beweiserfensters grün dargestellt.

`p2.key` Hier ist zu zeigen, dass $(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$ eine (aussagenlogische) Tautologie ist. Wenden Sie wieder Regeln an, indem Sie auf die Operatoren (`->`, `<->`, `!`) klicken. Die für Sie interessanten Regeln für die Aussagenlogik haben Namen, die sich aus dem Operator und der Seite der Sequenz zusammensetzen, also etwa `equiv_right` für eine Äquivalenz rechts vom Sequenzenpfeil und `impLeft` für eine Implikation links. Wenn Sie mit der

Mauszeiger etwas länger über dem Regelnamen verweilen, wird ein "Tooltip" angezeigt, der sagt, was eine Regelanwendung machen wird. Einige der Regelanwendungen werden zu Fallunterscheidungen („Case 1/2" im linken Teil des Fensters) führen. Durch Auswahl der Ziele im linken Teil können Sie auswählen, an welchem Sie arbeiten wollen. Letztlich müssen alle geschlossen werden. Wenn eine Formel links *und* rechts vom Sequenzenpfeil vorkommt, kann ein Ast geschlossen werden. Dazu wendet man bei der rechten Formel `close` an. Insgesamt entstehen vier zu schließende Äste.

`p3.key` Es geht um die prädikatenlogische Tautologie $(\exists x.\forall y.p(x,y)) \rightarrow (\forall v.\exists u.p(u,v))$. Überlegen Sie sich zunächst, warum diese Aussage gilt, und wie man sie beweisen könnte. Zerlegen Sie wieder die Formel, indem Sie bei den äußeren Operatoren anfangen. Wenn Sie einen $\exists$-Quantor links bzw. einen $\forall$-Quantor rechts abarbeiten (mit der Regel `exLeft` bzw. `allRight`), wird eine *Skolemkonstante* eingeführt, deren Name vom Namen der quantifizierten Variablen abgeleitet ist (z.B. x_0 für eine Quantifiziertung über x). Um einen $\forall$-Quantor links bzw. einen $\exists$-Quantor rechts mit einem Term t zu instanziieren, bietet die GUI mehrere Möglichkeiten:

- Man kann auf die quantifizierte Formel die Regel `allLeft` bzw. `exRight` anwenden. In dem sich öffnenden Instanziierungsdialog muss man den gewünschten Term t eingeben. Falls der Term bereits in der Sequenz vorkommt, kann man ihn auch per Drag&Drop in die Eingabemaske ziehen.
- Wenn der Term t bereits vorkommt, ist eine weitere Möglichkeit die Regel `instAll` bzw. `instEx`. Dazu klickt man *nicht* auf die quantifizierte Formel, sondern auf den Term t.
- Wenn der Term t bereits vorkommt, kann man als dritte Möglichkeit den Term einfach per Drag&Drop auf die zu instanziierende Variable ziehen.

`p4.key` Hier ist zu zeigen, dass $a = b \ \wedge \ b = c \rightarrow a = c$ eine Tautologie ist, wobei a, b, c irgendwelche Konstanten sind. Zerlegen Sie die Formel wieder mit `impRight` und `andLeft`. Sie können nun Gleichungen auf Terme anwenden. Um etwa die Gleichung `a=b` irgendwo anzuwenden, gibt es mehrere Möglichkeiten:

- Wählen Sie die Regel `make_insert_eq` auf dieser Gleichung. Es existiert nun eine Regel `insert_eq`, die für jedes Vorkommen des Terms `a` angeboten wird, und die diesen in `b` überführt.
- Alternativ können Sie auf ein Vorkommnis des Terms `a` klicken, und dort die Regel `applyEq` anwenden. Eventuell öffnet sich ein Instanziierungsdialog, in dem Sie `a=b` als die anzuwendende Gleichung auswählen müssen.
- Als dritte Möglichkeit können Sie die Gleichung `a=b` per Drag&Drop auf ein Vorkommnis des Terms `a` ziehen, und in dem sich öffnenden Kontextmenü die Regel `applyEq` auswählen.

Probieren Sie es aus! Wenn Sie eine Gleichung in der anderen Richtung, also von rechts nach links, anwenden wollen, können Sie sie mit `eqSymm` umdrehen, und dann wieder nach einer der beiden beschriebenen Möglichkeiten vorgehen.

`p5.key` Das Problem lautet $((\forall x.f(g(x)) = g(x)) \wedge (g(a) = b)) \rightarrow (f(b) = g(a))$. Es kann mit den bisher vorgestellten Techniken gelöst werden. überlegen Sie sich vorher, welche Instanz der all-quantifizierten Formel sie benötigen, und besorgen Sie sich diese mit einer der oben erwähnten Möglichkeiten, einen Allquantor zu instanziieren.

`p6.key` Überlegen sie sich, weshalb folgendes eine Tautologie ist

$$\forall x.\forall y.\forall z.(((P(x,y) \wedge P(y,z)) \rightarrow P(x,z)) \wedge \neg P(x,x)) \rightarrow \forall x.\forall y.(P(x,y) \rightarrow \neg P(y,x))$$

und beweisen Sie es danach im KeY-Beweiser.

`p7.key` Lesen Sie sich folgendes Rätsel aus dem Lufthansa-Magazin 11/2002 durch:

> Genau zwei der Personen A, B, C, D und E lügen. Welche?
> A: "B lügt dann und nur dann, wenn D die Wahrheit sagt!"
> B: "Sollte C ein wahrheitsliebender Mensch sein, so ist entweder A oder D ein Lügner."
> C: "E kann man auf keinen Fall trauen, und auch A oder B hält bzw. halten es mit der Wahrheit nicht so genau!"
> D: "Würde B die Wahrheit sagen, dann könnte man A oder C vertrauen!"
> E: "Unter den Personen A, C und D befindet sich mindestens ein Lügner!"

Überlegen Sie sich die Lösung des Problems und formalisieren Sie die Aufgabenstellung und Ihre Lösung als logische Formel. Beweisen Sie diese dann mit Hilfe von KeY. Sie können hierzu die Strategie *Java DL* verwenden!

Hinweis: Die zu beweisende Formel geben Sie im `\problem` Abschnitt von `p7.key` an. Die anderen Abschnitte in `p7.key` dienen der Deklaration der benötigten Signatur.

Abgabe bis 9.12.
Es braucht pro Gruppe nur *eine* Lösung abgegeben werden.
Die Abgabe der Übungsblätter erfolgt mit dem SVN System. Dazu legen Sie die abzugebenden Dateien im SVN ab und kopieren sie mit SVN in den Unterordner *abgabe/<nr>* wie in Aufgabe 2 auf Blatt 1 beschrieben.
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.

Praktikums-Webseite: `http://lfm.iti.uni-karlsruhe.de/keyprakt0809.php`
Christian Engel: R. 106, Tel. 608-4338, E-Mail: `engelc@ira.uka.de`
Benjamin Weiß: R. 309, Tel. 608-4324, E-Mail: `bweiss@ira.uka.de`