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 6

Aufgabe 13

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 aufg13.tgz
```

Sie erhalten 6 Dateien mit Namen `p1.key` bis `p6.key`. Jede dieser Dateien enthält eine Beweisaufgabe, die Sie mit dem KeY-Beweiser lösen sollen.

Starten Sie den KeY-Beweiser ohne TogetherCC, indem Sie (wenn Sie die Source-Code-Version intalliert haben) vom `bin/` Verzeichnis aus `runProver` aufrufen oder (wenn Sie die Byte-Code-Version intalliert haben) im `bin/` Verzeichnis `startProver` aufrufen.

Die Probleme sollen zunächst ohne Anwendung von Heuristiken gelöst werden. Wählen Sie den Menüpunkt `Options | Heuristics` und stellen Sie sicher, daß sich in der rechten Liste (aktive Heuristiken) keine Einträge befinden. Etwaige Einträge können mit einem Doppelklick in die linke Liste (deaktivierte Heuristiken) bewegt werden.

Problemdateien können jetzt mit dem Menüpunkt `File | Load` geladen werden. **Zur Abgabe** speichern Sie die Aufgaben mit `File | Save` als `p1.proof` bis `p6.proof`. 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. `close_by_true` 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 $(p \to q) \leftrightarrow (\neg q \to \neg p)$ 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 `imp_left` für eine Implikation links. 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_goal` an. Es wird ein Instantiierungsdialog angezeigt, der aber in diesen Aufgaben einfach mit dem „Apply“ Knopf quittiert werden kann. 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. Um einen $\forall$-Quantor links resp. einen $\exists$-Quantor rechts mit einem bereits in der Sequenz vorhandenen Term t zu instantiieren benutzt man am besten die `inst_all` resp. `inst_ex` Regel. Dazu klickt man *nicht* auf die quantifizierte Formel, sondern auf den Term t. Solange nur eine $\forall/\exists$ Formel für die Instantiierung in Frage kommt, kann man den erscheinenden Dialog wieder mit „Apply“ quittieren.

p4.key Hier ist zu zeigen, daß $a = b \wedge b = c \rightarrow a = c$, wobei a, b, c irgendwelche Konstanten sind. Zerlegen Sie die Formel wieder mit `imp_right` und `and_left`. Sie können nun Gleichungen auf Terme anwenden. Um etwa die Gleichung `a=b` irgendwo anzuwenden, wählen Sie die Regel `make_insert_eq` auf dieser Gleichung. Es existiert nun eine Regel `insert_eq` mit für jedes Vorkommen des Terms `a` angeboten wird, und die diesen in `b` überführt. Probieren Sie es aus! Wenn Sie eine Gleichung in der anderen Richtung, also von rechts nach links anwenden wollen, können Sie sie mit `commute_eq` umdrehen, um dann nochmals `make_insert_eq` anzuwenden.

p5.key Das Problem lautet $((\forall x.f(g(x)) = g(x)) \wedge (g(a) = b)) \rightarrow (f(b) = g(a))$. Das Problem 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 `inst_all`.

p6.key Das Problem gehört zu folgendem Rätsel (aus Lufthansa-Magazin 11/2002):

> 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!”

Schauen Sie sich die Formel in Problem 6 an. Welche Aussage wird dort formalisiert (Antwort als Zusammenfassung in einem Satz)? Beweisen Sie die Formel im KeY-Beweiser. Sie können hierzu die Heuristiken verwenden!

Aufgabe 14

Argumentieren Sie, weshalb folgendes eine Tautologie ist

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

Laden Sie die Datei `a15.key` in den Beweiser und zeigen Sie obige Tautologie.

Abgabe bis 19.1.2004

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`