Verfahren zur automatischen Verifikation

Prof. Dr. Christoph Walther • Markus Aderhold
FG Programmiermethodik • FB Informatik • TU Darmstadt
Sommersemester 2009

1. Übungsblatt

Übung am Dienstag, 28.04.2009

Hinweis Alle Verweise in den Aufgaben beziehen sich auf das Skriptum.

Aufgabe 1.1 ($\mathcal{FP}$-Programme)

(i) Definieren Sie eine $\mathcal{FP}$-Datenstruktur `structure bintree <= ...` für Binärbäume, in denen Schlüssel der Sorte `nat` in den inneren Knoten (und nicht in den Blättern wie bei der $\mathcal{FP}$-Datenstruktur `sexpr` aus Abschnitt 2.1) gespeichert werden.

(ii) Ein Suchbaum ist ein `bintree` b, so dass für jeden inneren Knoten k von b gilt: Der Schlüssel von k ist größer als alle Schlüssel im linken Teilbaum von k und der Schlüssel von k ist kleiner als alle Schlüssel im rechten Teilbaum von k. Suchbäume speichern *Mengen* von Schlüsseln, d.h. verschiedene Knoten eines Suchbaums besitzen verschiedene Schlüssel.

(ii.1) Schreiben Sie eine möglichst effiziente $\mathcal{FP}$-Prozedur

`function element(`n`:nat, `b`:bintree):bool <= ...`

die entscheidet, ob n als Schlüssel in einem Suchbaum b gespeichert ist. Dabei dürfen Sie für den Vergleich von Schlüsseln bzgl. "größer" und "kleiner" die vordefinierte $\mathcal{FP}$-Prozedur `function` $>$ aus Abschnitt 2.2 verwenden.

(ii.2) Schreiben Sie eine Prozedur

`function insert(`n`:nat, `b`:bintree):bintree <= ...`

so dass der als Ergebnis berechnete Binärbaum *genau* sowohl n als auch alle Schlüssel von b enthält, falls b ein Suchbaum ist. Weiter soll gelten, dass der Ergebnisbaum ein Suchbaum ist, falls der Eingabebaum b ebenfalls ein Suchbaum ist.

(iii) Geben Sie für alle Funktionssymbole Ihres $\mathcal{FP}$-Programms P aus den Teilaufgaben (i) und (ii) an, welche dieser Symbole Elemente von $\Sigma(P)^c$, $\Sigma(P)^d$, $\Sigma(P)^{proc}$, $\Sigma(P)^{rec}$ und $\Sigma(P)^{sel}$ sind.

Aufgabe 1.2 (Interpretation von $\mathcal{FP}$-Programmen)

(i) Geben Sie für die Prozedur `half` im Skript auf Seite 60 die Prozedurregel (2.8) des *Computation Calculus* (s. Definition 2.1) entsprechend Beispiel 2.3 an.

(ii) Berechnen Sie `half(5)` mit den Regeln des *Computation Calculus* (s. Definition 2.1) und geben Sie die Berechnung kommentiert wie in Beispiel 2.3 an.

Aufgabe 1.3 (Spezifikation von $\mathcal{FP}$-Programmen)

(i) Stellen Sie die Formel $(a \curvearrowright b) \curvearrowright (\neg b \curvearrowright \neg a)$ mit boolschen Variablen a und b als Term $t \in \mathcal{T}(\Sigma(P), \mathcal{V})_{bool}$ dar, indem Sie `if`-Ausdrücke für die logischen Junktoren verwenden.

(ii) Formulieren Sie die Forderungen für die Prozedur `insert` aus Aufgabe 1.1 als boolesche Terme mit implizit allquantifizierten Variablen. Sie dürfen dabei eine $\mathcal{FP}$-Prozedur

`function searchtree(`b`:bintree):bool <= ...`

voraussetzen, die entscheidet, ob b ein Suchbaum ist.

Aufgabe 1.4 (Semantik von $\mathcal{FP}$-Spezifikationen)

(i) Für die Prozeduren aus Aufgabe 1.1 sei folgendes $\mathcal{FP}$-Lemma gegeben:

```
lemma searchtree_1 <=
all b:bintree if(searchtree(b),true,searchtree(b))
```

Überprüfen Sie, ob dieses Lemma eine wahre Aussage nach Definition 3.1 ist, und beweisen Sie Ihre Behauptung.

(ii) Für die Prozeduren aus Aufgabe 1.1 sei folgendes $\mathcal{FP}$-Lemma gegeben:

```
lemma searchtree_2 <=
all b:bintree if(searchtree(b),searchtree(b),true)
```

Überprüfen Sie, ob dieses Lemma eine wahre Aussage nach Definition 3.1 ist, und beweisen Sie Ihre Behauptung.

(iii) Für die Prozeduren aus Aufgabe 1.1 sei folgendes $\mathcal{FP}$-Lemma gegeben:

```
lemma searchtree_3 <=
all b:bintree if(searchtree(b),searchtree(b),false)
```

Überprüfen Sie, ob dieses Lemma eine wahre Aussage nach Definition 3.1 ist, und beweisen Sie Ihre Behauptung.

(iv) Überprüfen Sie, ob Ihr Lemma aus Teilaufgabe 1.3(i) eine wahre Aussage nach Definition 3.1 ist, und beweisen Sie Ihre Behauptung.