WS 2015/2016 26.01.2016

Übungen zur Vorlesung
Bäume, Ordnungen und Anwendungen
Blatt 12

Prof. Dr. Roland Meyer Abgabe bis 02.02.2016 um 14h

Aufgabe 12.1 (Prüfungstermine)
Informieren Sie sich über mögliche Prüfungstermine, indem Sie die Website zur Vorlesung konsultieren.

Aufgabe 12.2 (Prädikatenabstraktion)
Zeigen Sie folgende Aussagen.

a) Bei $\overline{q_1 \vee q_2}$ handelt es sich tatsächlich um $q_1 \sqcup q_2$ in $\mathrm{Abs}(P)$.

b) Es gilt $\bigwedge\{q \in \mathrm{Abs}(P) \mid b \models q\} \models\!\!\models \bigwedge\{l \in P \vee \neg P \mid b \models l\}$ für beliebiges b.

Aufgabe 12.3 (Weakest Preconditions)

a) Zeigen Sie folgende Äquivalenz:

$$\begin{aligned} & wp(\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}, p) \\ \models\!\!\models\ & wp(\texttt{if } * \texttt{ then assume } b;\ c_1 \texttt{ else assume } \neg b;\ c_2 \texttt{ end}, p), \end{aligned}$$

wobei

$$\begin{aligned} wp(\texttt{assume } b, p) &:= p \vee \neg b \\ wp(\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}, p) &:= (b \to wp(c_1, p)) \wedge (\neg b \to wp(c_2, p)) \\ wp(\texttt{if } * \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}, p) &:= (\mathit{true} \to wp(c_1, p)) \wedge (\mathit{true} \to wp(c_2, p)) \end{aligned}$$

gilt. Bei `if` $*$ handelt es sich um eine nicht-deterministische Auswahl. Diese Transformation ist beim Erzeugen von Gegenbeispielen hilfreich, da man bei jeder bedingten Anweisung erkennen kann, welcher Ausführungszweig gewählt wurde.

b) Übertragen Sie obiges Prinzip auf `while` Schleifen und geben Sie eine entsprechende Äquivalenz an.

c) Berechnen Sie $wp(com, \mathit{false})$, wobei com definiert ist als:

$$\begin{aligned} com \equiv\ & z := 0;\ \texttt{assume } x > 0;\ \texttt{assume } y > 0; \\ & \texttt{assume } \neg(x > 0);\ \texttt{assume } z = 0;\ \texttt{skip}; \end{aligned}$$

Aufgabe 12.4 (Control-State-Reachability)
Eine *Safety-Eigenschaft* oder *Invariante* eines Programms ist eine Menge **Safe** $\subseteq$ **Prog** $\times$ **State** von Konfigurationen. Man sagt, die Eigenschaft ist *erfüllt*, falls alle erreichbaren Konfigurationen in **Safe** liegen, anderenfalls ist sie *verletzt*. Beispiele für Safety-Eigenschaften sind: „Es gibt keine Divisionen durch 0“ oder „Die Variable x ist immer positiv“.
Erklären Sie, wie man die Überprüfung von Safety-Eigenschaften eines Programms auf einen Erreichbarkeitscheck für einen Kontrollzustand c_{bad} reduzieren kann. Machen Sie Ihre Vorgehensweise an einem der obigen Beispiele deutlich.

Abgabe bis 02.02.2016 um 14h im Kasten neben Raum 34-401.4