

Aufgabe 5-1

Beweisen Sie die Gültigkeit der folgenden Formeln im Sequenzen-Kalkül:

- a) $(\forall x.P(x)) \Rightarrow (\exists x.P(x))$
- b) $(\exists x.(P(x) \vee Q(x))) \Rightarrow (\exists x.P(x)) \vee (\exists x.Q(x))$

Die folgende Formel ist nicht gültig. Erklären Sie, woran ein Sequenzen-Beweis scheitert.

- c) $\exists x.P(x) \wedge \exists x.Q(x) \Rightarrow \exists x.(P(x) \wedge Q(x))$

Aufgabe 5-2

Beweisen Sie folgenden Formeln mit Hilfe der Tableaumethode:

- a) $(\forall x.\neg P(x)) \Rightarrow \neg(\exists x.P(x))$
- b) $(\forall x.P(x) \vee Q(x)) \Rightarrow (\exists x.P(x) \vee \forall x.Q(x))$

Aufgabe 5-3

Diese Aufgabe wiederholt noch einmal einfache aussagenlogische und prädikatenlogische Beweise in PVS.

Auf der Webseite zur Vorlesung

<http://www.informatik.uni-ulm.de/ki/maschinelles-beweisen.html>

findet sich die Datei `basic.pvs`. Beweisen Sie eine Auswahl der dort aufgeführten Formeln!

Aufgabe 5-4

In dieser Aufgabe modellieren wir einfache elektronische Bausteine in PVS. Mit einem *Halbaddierer* können zwei einstellige Binärzahlen (Bits) addiert werden. Er besteht aus zwei Eingängen x und y sowie den beiden Ausgängen s und c_{out} . Dabei liefert s die (rechte Stelle der) Summe der Eingabe-Bits und c_{out} den Übertrag.

- a) Definieren Sie in PVS zwei Boolesche Funktionen `HA_s` und `HA_cout` vom Typ `[bool, bool -> bool]` die die betreffenden Ausgabewerte eines Halbaddierers berechnen. (Stellen Sie ggf. zunächst eine Wahrheitstafel auf und lesen Sie daraus die Gleichungen für `s` und `cout` ab.)
- b) Definieren Sie in PVS eine Funktion `bool2nat` die einen (Booleschen) Bit-Wert in die entsprechende natürliche Zahl umrechnet.
- c) Formulieren Sie ein Korrektheitstheorem für Ihren Halbaddierer: Ein Halbaddierer ist korrekt, wenn die durch die Ausgänge repräsentierte natürliche Zahl die Summe der (als natürliche Zahl interpretierten) Eingangswerte ist. Beweisen Sie das Theorem.

Ein Volladdierer (engl. *full adder*) ist eine Erweiterung des Halbaddierers. Ein Volladdierer besitzt zusätzlich zu den beiden Eingängen `x` und `y` einen dritten Eingang `cin`, der den Übertrag aus der vorigen Stelle beinhaltet. Wie beim Halbaddierer liefern die beiden Ausgänge `s` und `cout` die (rechte Stelle der) Summe der (jetzt drei) Eingabe-Bits bzw. den Übertrag.

- d) Modellieren Sie in ähnlicher Weise wie oben einen Volladdierer in PVS und beweisen Sie ein entsprechendes Korrektheitstheorem.

neue PVS-Befehle: (EXPAND), (EXPAND*), (LIFT-IF), (IFF), (ASSERT), evtl. (BETA).

Erweiterungsmöglichkeiten:

- e) (Optional) Anstatt die beiden Ausgänge der Addierer einzeln durch jeweils eine Funktion zu modellieren, kann auch eine Funktion definiert werden, die die beiden Ausgabewerte zusammen als ein *Paar* liefert. Modellieren Sie entsprechende Funktionen für den Halb- bzw. den Volladdierer. Beweisen Sie auch die Korrektheit Ihrer Definitionen.
(In PVS kann mit Hilfe der *Projektionen* `p'1`, `p'2`, `p'3`, usw. auf die jeweiligen Komponenten eines Tupels zugegriffen werden.)
- f) (Optional) Ein Volladdierer kann auch aus zwei Halbaddierern und einem ODER-Gatter aufgebaut werden. Definieren Sie in dieser Weise eine zweite Version eines Volladdierers und zeigen Sie, dass die beiden Varianten gleich sind.
- g) (Optional) Definieren Sie eine Additionsfunktion `add`, die zwei Binärzahlen mit fest vorgegebener Stelligkeit n (z. B. $n = 4$) mit Hilfe n Volladdierern addiert. Beweisen Sie die Korrektheit dieses Addierers. (Welche Modellierungs- und Beweisverfahren würde für beliebige n benötigt werden?)