Institut für Informatik
der Universität München
Prof. M. Hofmann
A. Abel

WS 2005/2006

12. Januar 2006

Übungen zur Vorlesung
Rechnergestütztes Beweisen

Blatt 11

Aufgabe 28: [Papier, 5 Punkte] Betrachten Sie das in der Vorlesung behandelte Zustandsübergangssystem *Semaphore*. Drücken Sie folgende Aussagen in CTL aus:

1. Immer wenn Prozess 1 wartet, ist Prozess 2 kritisch.
2. Wenn Prozess 1 wartet und Prozess 2 untätig ist, ist Prozess 1 im nächsten Zustand kritisch.
3. Wenn zu Beginn beide Zustände untätig sind und die Ampel auf *frei* steht, so gilt Aussage 2 irgendwann einmal.
4. Unter der Vorbedingung von Aussage 3 gilt, dass einer der beiden Prozesse unendlich oft kritisch ist.
5. Unter der Vorbedingung von Aussage 3 gilt, dass Prozess 1 unendlich oft kritisch ist.

Entscheiden Sie, welche Aussagen wahr sind und welche falsch.

Aufgabe 29: [Papier, 6 Punkte] Sei M eine Menge und $\Phi : M \to M$ ein monotoner Operator auf M, d.h., $P \subseteq Q$ impliziert $\Phi(P) \subseteq \Phi(Q)$. Iteration von Φ sei definiert durch $\Phi^0(Q) = Q$, $\Phi^{n+1}(Q) = \Phi(\Phi^n(Q))$. Gegeben sind zwei Kandidaten für den kleinsten Fixpunkt von Φ:

$$A := \bigcup\{\Phi^i(\emptyset) \mid i \in \mathbb{N}\} \qquad \text{und} \qquad B := \bigcap\{Q \subseteq M \mid \Phi(Q) \subseteq Q\}.$$

Beweisen Sie:

1. $A \subseteq B$. (Hinweis: Zeigen Sie zuerst $\Phi(B) \subseteq B$.)
2. Falls M endlich, dann auch $B \subseteq A$. (Hinweis: Zeigen Sie zuerst, dass die Folge $\Phi^n(\emptyset)$ stationär wird, d.h., $\Phi^{n_0+i}(\emptyset) = \Phi^{n_0}(\emptyset)$ für ein n_0. Dann zeigen Sie $\Phi(A) \subseteq A$.)

Aufgabe 30: [PVS, 9 Punkte] Das *Alternating Bit Protocol* beschreibt eine Methode, Daten über eine Verbinding zu übertragen, die Pakete verlieren oder duplizieren kann (allerdings werden keine Pakete verändert): Der Sender markiert die Nachricht mit einem Bit, und der Empfänger weiss, welche Markierung er zu erwarten hat. Empfängt er nun ein Paket mit der richtigen Markierung, schickt er diese Markierung an den Sender zurück und erwartet in Zukunft ein Paket mit der anderen Markierung. Der Sender, sobald er vom Empfänger die Markierung erhält, mir der er auch das ausgehende Paket versehen hatte, schickt ein neues Paket, diesmal mit der anderen Markierung, usw.

Wir modellieren das Protokoll durch die vier Entitäten *Sender*, *Kanal1*, *Empfänger* und *Kanal2*. Die Nachricht werde der Einfachheit halber auch durch ein Bit modelliert. Das heisst, dass Kanal1 (vom Sender zum Empfänger) insgesamt 2 Bit transportiert: die Nachricht, `ch_msg`, und die Markierung, `ch_tag`. Kanal2 (Rückrichtung) transportiert nur ein Markierungs-Bit, `ch_ack`. Der Sender wird durch 3 Bit modelliert: `sending` (gerade auf Sendung), `msg` (die zu sendende Nachricht) und `tag` (die zu sendende Markierung). Der Empfänger ist ebenfalls durch 3 Bit beschreibbar: `receiving` (gerade auf Empfang), `expected` (die erwartete Markierung) und `ack` (die zuletzt empfangene Markierung).

Zu Beginn sind Sender und Empfänger beide aktiv, `tag` und `expected` auf 1 und `ch_tag`, `ack` und `ch_ack` auf 0 gesetzt.

Nun sind folgende Aktionen möglich:

- Ist der Sender nicht aktiv, ändert er die zu sendende Markierung, wählt sich nicht deterministisch eine neue Nachricht und wird aktiv.
- Stimmt die Markierung auf Kanal1 nicht mit dem Sender überein, übernimmt Kanal1 Markierung und Nachricht vom Sender.
- Ist der Empfänger aktiv und stimmt die Markierung auf Kanal1 mit der erwarteten Markierung überein, so wird der Empfänger inaktiv.
- Ist der Empfänger inaktiv, ändert er die erwartete Markierung und setzt `ack` auf die empfangene Markierung.
- Ist die Markierung auf Kanal2 ungleich `ack`, übernimmt Kanal2 diesen Wert.
- Ist der Sender aktiv und stimmt die Markierung auf Kanal2 mit der gesendeten überein, so wird der Sender inaktiv.

Das Protokoll arbeitet in folgendem Falle korrekt: Immer, wenn der Sender inaktiv wird, so ist die empfangene Nachricht, also die sich auf Kanal1 befindet, gleich der gesendeten.

1. Formalisieren Sie das Protokoll in PVS.
2. Beweisen Sie, dass in allen erreichbaren Zuständen die Korrektheitseigenschaft gilt.
3. Beweisen Sie, dass bei unendlicher Laufzeit auch unendlich viele Bits übertragen werden.
4. Testen Sie ihre Formalisierung, indem Sie ein paar falsche Aussagen zu beweisen suchen.

Hinweis: falls Sie alles richtig formalisiert haben sind die Beweise Einzeiler: `(model-check)`.

Abgabe: bis **Dienstag, 17.01., 18.00 Uhr** im Sekretariat (**Z1.05**) oder bei Andreas Abel im Büro (**D1.09**). Elektronische Lösungsteile per Mail an `abel@informatik.uni-muenchen.de`.