**Ludwig-Maximilians-Universität München**
**Institut für Informatik**
Prof. Dr. F. Kröger, M. Hammer

WS 2006/2007
Übungsblatt 11

# Temporale Logik und Zustandssysteme Lösungsvorschlag

**Aufgabe 11-1** **Stapel** (8 Punkte)

In dieser Aufgabe soll ein Stapel (Stack, FILO-Buffer) natürlicher Zahlen temporallogisch modelliert werden.

Die der Modellierung zugrundeliegende Signatur SIG verfügt über die Sorten, $\mathsf{S}$ und $NAT$ und $NAT^\bullet$ mit folgender Interpretation:

- $NAT$ wird durch $\mathbb{N}$ interpretiert (Objekte, die auf den Stapel gelegt werden).
- $NAT^\bullet$ wird interpretiert durch die Menge $\mathbb{N} \cup \{\bullet\}$ (Das Symbol "$\bullet$" signalisiert einen Fehler).
- $STACK$ (der Stapel) wird interpretiert durch die Menge $\mathbb{N}^*$ der endlichen Folgen natürlicher Zahlen.

Eine endliche Folge $(n_0, n_1, \ldots, n_k)$ von natürlichen Zahlen entspricht einem Stapel, auf den die Zahlen $n_0, n_1, \ldots, n_k$ in dieser Reihenfolge abgelegt wurden.

Die Signatur enthält ferner

- die Individuenkonstante $\text{EMPTY}^{(\epsilon, STACK)}$, interpretiert durch die leere Folge.
- das Funktionssymbol $\text{APPEND}^{(NAT\ STACK, STACK)}$, interpretiert durch die Funktion $((n_0, \ldots, n_k), n) \mapsto (n_0, \ldots, n_k, n)$, die eine Zahl hinten an eine Folge anfügt.
- das Funktionssymbol $\text{LAST}^{(STACK, NAT^\bullet)}$, interpretiert durch die Funktion, die das letzte Element einer Folge zurückliefert, bzw. $\bullet$ für die leere Folge.
- das Funktionssymbol $\text{LEN}^{(STACK, NAT)}$, interpretiert durch die Funktion, die die Länge einer Folge natürlicher Zahlen bestimmt.

Wir betrachten die Aktionenmenge $Act = \{\mathsf{push}, \mathsf{pop}\}$. Ein Ausführen der Aktion $\mathsf{push}$ steht hier für das Ablegen einer Zahl auf dem Stapel, die Aktion $\mathsf{pop}$ bedeutet das Herunternehmen einer Zahl.

Beachten Sie, dass nur dann eine Zahl heruntergenommen werden kann, wenn der Stapel mindestens eine Zahl enthält.

a) Geben Sie ein markiertes Transitionssystem $(X, V, Z, T, Act, \mathcal{E})$ an, das das Verhalten des Stapels modelliert. Ihre Modellierung soll mindestens eine Systemvariable $s$ enthalten, die den Zustand des Stapels repräsentiert.

**Lösung:** Wir setzen $Z_1 = \{\eta : X \cup V \to \mathbb{N}^* \cup \{\textbf{true}, \textbf{false}\} \mid \eta(s) \in \mathbb{N}^*, \eta(exec\,\mathsf{push}), \eta(exec\,\mathsf{pop}) \in \{\textbf{true}, \textbf{false}\}\}$. Wir definieren das Transitionssystem $(X, V, Z, T, Act, \mathcal{E})$ wie folgt:

$X = \{s\} \qquad V = \{exec\,\mathsf{push}, exec\,\mathsf{pop}\}$

$Z = \{\eta \in Z_1 \mid \eta(exec\,\mathsf{push}) = \mathsf{tt} \iff \eta(exec\,\mathsf{pop}) = \mathsf{ff} \text{ und } \eta(s) \neq \text{EMPTY}^{\mathsf{S}} \Rightarrow \eta(exec\,\mathsf{pop}) = \textbf{false}\}$

$\mathcal{E} = \{\text{enabled}_{\mathsf{push}}, \text{enabled}_{\mathsf{pop}}\}$ mit $\text{enabled}_{\mathsf{push}} \equiv \textbf{true}$ und $\text{enabled}_{\mathsf{pop}} \equiv s \neq \text{EMPTY}$

mit der Zustandsübergangsrelation $T \subseteq Z \times Z$ definiert durch $(\eta, \eta') \in T$ genau dann wenn

- $\eta(s) \neq \text{EMPTY}^{\mathsf{S}}, \eta(s) = \text{APPEND}^{\mathsf{S}}(\text{LAST}^{\mathsf{S}}(\eta(s)), \eta'(s))$ und $\eta(exec\,\mathsf{pop}) = \mathsf{tt}$, oder
- $\eta'(s) = \text{APPEND}^{\mathsf{S}}(x, \eta(s))$ für ein $x \in \mathbb{N}$ und $\eta(exec\,\mathsf{push}) = \mathsf{tt}$.

b) Geben Sie eine Menge $\mathcal{A}$ von temporallogischen Formeln an, die das Verhalten des Stapels möglichst genau beschreiben.

**Lösung:** Wir wählen die folgenden Formeln:

- (nnil) $\neg\mathrm{nil}$
- (push) $exec\,\mathsf{push} \rightarrow \exists n.s' = \textsc{append}(s, n)$
- (pop) $exec\,\mathsf{pop} \rightarrow s \neq \textsc{empty} \wedge \exists n.s = \textsc{append}(s', n)$
- (pp) $exec\,\mathsf{push} \leftrightarrow \neg exec\,\mathsf{pop}$

c) Formulieren Sie die Eigenschaft

> "Jede Zahl, die vom Stapel genommen wurde, wurde irgendwann einmal auf den Stapel gelegt"

als temporallogische Formel $F$ in der Sprache $\mathcal{L}^p_{\mathrm{FOLTL}}$. *Hinweis:* Formulieren Sie zunächst die Eigenschaft "die Zahl $n$ wurde auf den Stapel gelegt" als temporallogische Formel.

**Lösung:** Wir setzen zunächst $A(n) \equiv exec\,\mathsf{push} \wedge s' = \textsc{append}(s, n)$. Dann ist $A(n)$ ein Stapel, auf den im letzten Schritt die Zahl $n$ (mittels push) gelegt wurde. Die gewünschte Eigenschaft ist nun $F \equiv \textsc{last}(s) = n \rightarrow \Diamond\!\!\!-\, A(n)$.

d) Geben Sie einen Ablauf $(\eta_0, \eta_1, \dots)$ des Transitionssystems an, in dem $F$ *nicht* gilt.

**Lösung:** Wir setzen $\eta_0(s) = (15)$, ansonsten beliebig. Die Idee ist, mit einem nicht-leeren Stapel zu starten.

e) Formulieren Sie die Eigenschaft

> "Wenn der Stapel leer ist, so gilt von nun an die Formel $F$"

als temporallogische Formel $G$.

**Lösung:** Wir setzen $G \equiv s = \textsc{empty} \rightarrow \Box F$.

f) Zeigen Sie: In jedem Ablauf des Transitionssystems gilt die Formel $G$.

**Lösung:** Sei $(\eta_0, \dots, )$ ein Ablauf des Transitionssytems und $i \in \mathbb{N}$ mit $\eta_i(s) = \textsc{empty}^{\mathsf{S}}$.

Wir definieren induktiv

$$M_0 = \emptyset \text{ und } M_{i+1} = \begin{cases} M_i & \text{, falls } \textsc{len}^{\mathsf{S}}(\eta_{i+1}(s)) \leq \textsc{len}^{\mathsf{S}}(\eta_i(s)) \\ M_i \cup \{n\} & \text{, falls } \eta_{i+1}(s) = \textsc{append}^{\mathsf{S}}(\eta_i(s), n) \end{cases}.$$

$M_i$ ist die Menge der Elemente, die zum Zeitpunkt $i$ auf den Stapel gelegt wurden. Für eine Folge $(n_0, \dots, n_k)$ sei $\cup(n_0, \dots, n_k) = \{n_0, \dots, n_k\}$. Wir zeigen nun, dass $\cup\eta_j(s) \subseteq M_j$ für alle $j \geq i$. Für $j = i$ ist dies klar. Für $j > i$ und $\eta_{j-1}(exec\,\mathsf{pop}) = \mathsf{tt}$ folgt dies aus der Induktionsvoraussetzung. Für $\eta_{j-1}(\mathsf{push}) = \mathsf{tt}$ haben wir $M_j = M_{j-1} \cup \{n\}$, wobei $n$ das neue oberste Element ist, nach Definition von $M_j$, also $n \in M_j$. Für $n' \in \cup\eta_j(s)$ und $n' \neq n$ folgt dies aus der Induktionsvoraussetzung.

Nach Definition von $exec\,\mathsf{pop}$ gilt damit die Eigenschaft.

g) Zeigen Sie: $\mathcal{A} \vdash \textsc{len}(s) = n \wedge exec\mathsf{push} \rightarrow \textsc{len}(s) > n$ **unless** $exec\mathsf{pop}$ durch Angabe einer Herleitung. Die Gesetze (T1)-(T59) sowie die abgeleitete Regel

$$\begin{aligned} (\mathrm{unless}_\Gamma) \quad & exec\lambda \wedge A \rightarrow \circ C \vee \circ(B \wedge A) \quad \text{für alle } \lambda \in Act \\ & \mathrm{nil}_\Gamma \wedge A \rightarrow B \vee C \\ & \vdash A \rightarrow B \textbf{ unless } C \end{aligned}$$

dürfen verwendet werden.

**Lösung:** Wir setzen $A \equiv \text{LEN}(s) \geq n \wedge \mathit{exec}\,\mathsf{push}$, $B \equiv \text{LEN}(s) > n$ und $C \equiv \mathit{exec}\,\mathsf{pop}$ und erhalten folgende Herleitung:

| | | |
|---|---|---|
| (1) | $\mathit{exec}\,\mathsf{pop} \wedge A \rightarrow \textbf{false}$ | (prop)(pp) |
| (2) | $\mathit{exec}\,\mathsf{pop} \wedge A \rightarrow \circ C \vee \circ(B \wedge A)$ | (prop)(1) |
| (3) | $s' = \text{APPEND}(s, n) \rightarrow \text{LEN}(s') > \text{LEN}(s)$ | (data) |
| (4) | $\mathit{exec}\,\mathsf{push} \rightarrow \text{LEN}(s') > \text{LEN}(s)$ | (push)(3) |
| (5) | $\mathit{exec}\,\mathsf{push} \wedge A \rightarrow \circ(\text{LEN}(s) > n)$ | (pred)(4) |
| (6) | $\circ(\mathit{exec}\,\mathsf{push} \vee \mathit{exec}\,\mathsf{pop})$ | (pp) |
| (7) | $\mathit{exec}\,\mathsf{push} \wedge A \rightarrow \circ C \vee \circ(B \wedge A)$ | (prop)(5)(6) |
| (8) | $\text{nil} \wedge A \rightarrow B \vee C$ | (nnil)(prop) |
| (9) | $A \rightarrow B$ **unless** $C$ | (unless)(2)(7)(8) |
| (10) | $\text{LEN}(s) = n \wedge \mathit{exec}\,\mathsf{push} \rightarrow \text{LEN}(s) \geq n \wedge \mathit{exec}\,\mathsf{push}$ | (prop) |
| (11) | $\text{LEN}(s) = n \wedge \mathit{exec}\,\mathsf{push} \rightarrow B$ **unless** $C$ | (prop)(9)(10) |

**Aufgabe 11-2** **Warteschlange** (6 Punkte)

Gegeben seien die Sorten $QUEUE$ und $NAT$, die Funktionszeichen ENQ und DEQ sowie die Prädikatszeichen IN und ISEMPTY mit folgenden intuitiven Bedeutungen. Die Elemente (der Interpretation von) $QUEUE$ sind endliche Folgen von natürlichen Zahlen mit den für "Schlangen" üblichen Operationen ENQ und DEQ sowie den Prädikaten ISEMPTY und IN:

- $\text{ENQ}(q, n)$ ist die Schlange, die entsteht, wenn man die natürliche Zahl $n$ an die Schlange $q$ hinten anfügt.
- $\text{DEQ}(q)$ ist die Schlange, die entsteht, wenn man das vorderste Element von $q$ entfernt, falls $q$ nicht leer ist, und die leere Schlange sonst.
- $\text{ISEMPTY}(q)$ ist wahr genau dann, wenn $q$ die leere Schlange ist.
- $\text{IN}(n, q)$ ist wahr genau dann, wenn $n$ in $q$ enthalten ist.

Weiter seien $N$ eine Individuenkonstante und $x$ eine Variable der Sorte $NAT$.

Sei $\Pi$ das folgende PAR-Programm (mit durch vorstehende Angaben informell beschriebener Signatur $SIG_\Pi$ und Struktur $\mathsf{S}_\Pi$):

$\Pi \equiv$ **var** $q : QUEUE; n : NAT$
**start** $\text{ISEMPTY}(q) \wedge n > N$
**cobegin loop** $\alpha_0 : q := \text{ENQ}(q, n);$
$\alpha_1 : n := n + 1$
**endloop**
$\|$
**loop** $\beta$ : **await** $\neg\text{ISEMPTY}(q)$ **then** $q := \text{DEQ}(q)$
**endloop**
**coend**

a) Beweisen Sie durch Angabe einer Herleitung: $\mathcal{A}_\Pi \vdash \Box(n > N)$.

**Lösung:**

| | | |
|---|---|---|
| (1) | $start_\Pi \rightarrow n > N$ | (prop) |
| (2) | $exec\alpha_0 \rightarrow n' = n$ | (ASSIGN) |
| (3) | $exec\alpha_0 \wedge n > N \rightarrow \circ(n > N)$ | (pred)(2) |
| (4) | $n > N$ **invof** $\alpha_0$ | (3) |
| (5) | $n' = n + 1 \wedge n > N \rightarrow n' > N'$ | (data) |
| (6) | $exec\alpha_1 \rightarrow n' = n + 1$ | (ASSIGN) |
| (7) | $exec\alpha_1 \wedge (n > N) \rightarrow \circ(n > N)$ | (5)(6)(pred) |
| (8) | $n > N$ **invof** $\alpha_1$ | (7) |
| (9) | $exec\beta \rightarrow n' = n$ | (ASSIGN) |
| (10) | $exec\beta \wedge n > N \rightarrow \circ(n > N)$ | (9)(pred) |
| (11) | $n > N$ **invof** $\beta$ | (10) |
| (12) | $n > N$ **invof** $Act_\Pi$ | (4)(8)(11) |
| (13) | $\Box(n > N)$ | (invstart$_\Pi$)(1)(12) |

b) Beweisen Sie durch Angabe einer Herleitung: $\mathcal{A}_\Pi \vdash \Box(\text{IN}(x, q) \rightarrow x > N)$.

**Lösung:** Sei $I \equiv \text{IN}(x, q) \rightarrow x > N$.

| | | |
|---|---|---|
| (1) | $start_\Pi \rightarrow \text{ISEMPTY}(q)$ | (prop) |
| (2) | $start_\Pi \rightarrow \neg\text{IN}(x, q)$ | (prop)(1)(H1) |
| (3) | $start_\Pi \rightarrow (\text{IN}(x, q) \rightarrow x > N)$ | (prop)(2) |
| (4) | $\text{IN}(x, \text{ENQ}(n, q)) \rightarrow x = n \vee \text{IN}(x, q)$ | (H2)(prop) |
| (5) | $exec\alpha_0 \rightarrow q' = \text{ENQ}(q, n)$ | (ASSIGN) |
| (6) | $exec\alpha_0 \rightarrow n > N$ | (a))(ltl3)(prop) |
| (7) | $exec\alpha_0 \wedge I \rightarrow (\text{IN}(x, q') \rightarrow x = n \vee \text{IN}(x, q))$ | (4)(5)(prop) |
| (8) | $exec\alpha_0 \wedge I \rightarrow (\text{IN}(x, q') \rightarrow x > N)$ | (6)(7)(pred) |
| (9) | $I$ **invof** $\alpha_0$ | (8) |
| (10) | $I$ **invof** $\alpha_1$ | (ASSIGN) |
| (11) | $exec\beta \wedge I \rightarrow q' = \text{DEQ}(q)$ | (ASSIGN) |
| (12) | $exec\beta \wedge I \rightarrow (\text{IN}(x, q') \rightarrow \text{IN}(x, q))$ | (H3)(prop)(11) |
| (13) | $exec\beta \wedge I \rightarrow (\text{IN}(x, q') \rightarrow x > N)$ | (prop)(12) |
| (14) | $I$ **invof** $\beta$ | (13) |
| (15) | $I$ **invof** $Act_\Pi$ | (9)(10)(14) |
| (16) | $\Box I$ | (invstart$_\Pi$)(3)(15) |

c) Geben Sie eine Formel aus $\mathcal{L}_{\text{TL}\Pi}$ an, die besagt, dass jedes Element, das in der Schlange enthalten ist, irgendwann „ganz vorne steht“.

**Lösung:** Informell: Wenn $x$ in der Schlange steht, dann ist es irgendwann das Element, daß aus der Schlange verschwindet, wenn das Kopfelement entnommen wird:

$$\forall x \text{IN}(x, q) \rightarrow \Diamond(\text{IN}(x, q) \wedge \neg\text{IN}(x, \text{DEQ}(q)))$$

Hinweis: Für die Herleitungen sind insbesondere folgende (data)-Axiome nützlich:

(H1) $\text{ISEMPTY}(q) \rightarrow \neg\text{IN}(x, q)$.

(H2) $\text{IN}(x, \text{ENQ}(n, q)) \leftrightarrow x = n \vee \text{IN}(x, q)$.

(H3) $\text{IN}(x, \text{DEQ}(q)) \rightarrow \text{IN}(x, q)$.

**Aufgabe 11-3** **Mengen** (keine Abgabe)

Das folgende PAR-Programm vertauscht Elemente der endlichen Mengen $S$ und $T$ von natürlichen Zahlen.

$\Pi \equiv$ **var** $min, max, min_0, max_0 : NAT^\infty$;
$S, T : NATS$
**start** $max_0 > \max(T) \wedge min_0 < \min(S) \wedge min = min_0 \wedge max = max_0$
**cobegin loop** $\alpha_0 : min := \min(S)$;
$\alpha_1$ : **await** $min < max \wedge max < max_0$ **then** $max_0 := max$;
$\alpha_2 : S := (S \setminus \{min\}) \cup \{max_0\}$
**endloop**
$\|$
**loop** $\beta_0 : max := \max(T)$;
$\beta_1$ : **await** $max > min \wedge min > min_0$ **then** $min_0 := min$;
$\beta_2 : T := (T \setminus \{max\}) \cup \{min_0\}$
**endloop**
**coend**

Die informellen Bedeutungen der Sorten $NAT^\infty$ und $NATS$ sind: (die Interpretation von) $NAT^\infty$ enthält alle natürlichen Zahlen sowie die Pseudowerte $\infty$ und $-\infty$; die Relationen $<$, $>$, usw. sind wie üblich erweitert auf $NAT^\infty$, d.h. für alle natürlichen Zahlen $n$ gilt $-\infty < n$ und $n < \infty$.
Die Sorte $NATS$ besteht aus allen endlichen Mengen von natürlichen Zahlen. Die Interpretationen von $\cup$, $\setminus$ sind wie üblich, $\min(U)$ bzw. $\max(U)$ bezeichen das kleinste bzw. größte Element einer Menge $U$, falls diese nicht leer ist, und sonst die Pseudowerte $-\infty$ bzw. $\infty$. Insbesondere gelten also u.a. folgende Daten-Axiome:

$$x \in U \rightarrow x \geq \min(U),$$
$$x \in U \rightarrow x \leq \max(U),$$
$$\min(U) = k \wedge x \geq k \rightarrow \min((U \setminus \{k\}) \cup \{x\}) \geq k.$$

a) Beweisen Sie $\mathcal{A}_\Pi \vdash start_\Pi \rightarrow \Box I$ für

$$\begin{aligned} I \equiv\ & \wedge (max_0 \geq max) \wedge (min_0 \leq min) \\ & \wedge (min \leq \min(S)) \wedge (at\{\alpha_1, \alpha_2\} \rightarrow min = \min(S)) \\ & \wedge (max \geq \max(T)) \wedge (at\{\beta_1, \beta_2\} \rightarrow max = \max(T)) \\ & \wedge (at\,\alpha_2 \rightarrow min \leq \max_0) \wedge (at\,\beta_2 \rightarrow min_0 \leq max), \end{aligned}$$

wobei $at\,M$ für eine endliche Menge $M$ von Marken die Formel $\bigvee_{\alpha \in M} at\,\alpha$ bezeichnet.

Sie dürfen ohne Beweis voraussetzen, dass $\mathcal{A}_\Pi \vdash I$ **invof** $\{\beta_0, \beta_1, \beta_2\}$ gilt.

**Lösung:** Es stehe $at\{\alpha_1, \ldots, \alpha_n\}$ für $\bigvee_{i=1}^{n} at\alpha_i$ (analog für **invof**).

| | | |
|---|---|---|
| (1) | $start_\Pi \rightarrow at\alpha_0 \wedge at\beta_0 \wedge max_0 > \max(T) \wedge min_0 < \min(S)$ $\wedge min = min_0 \wedge max = max_0$ | (pred) |
| (2) | $at\alpha_0 \rightarrow \neg at\{\alpha_1, \alpha_2\}$ | (PC) |
| (3) | $at\beta_0 \rightarrow \neg at\{\beta_1, \beta_2\}$ | (PC) |
| (4) | $start_\Pi \rightarrow I$ | (1)(2)(3)(data) |
| (5) | $I \rightarrow min_0 \leq \min(S)$ | (data) |
| (6) | $exec_{\alpha_0} \wedge I \rightarrow \circ(min_0 \leq min)$ | (5)(ASSIGN)(data)(pred) |
| (7) | $exec_{\alpha_0} \rightarrow \circ(at\{\alpha_1, \alpha_2\} \rightarrow min = \min(S))$ | (ASSIGN)(pred) |
| (8) | $exec_{\alpha_0} \rightarrow \circ\neg at\alpha_2$ | (C1)(C3)(prop) |
| (9) | $I$ **invof** $exec_{\alpha_0}$ | (ASSIGN)(6)(7)(8) |
| (10 | $exec_{\alpha_1} \wedge I \rightarrow \circ(max_0 \geq max)$ | (ASSIGN)(data) |
| (11) | $exec_{\alpha_1} \rightarrow min \leq max$ | (action) |
| (12) | $exec_{\alpha_1} \rightarrow \circ(min \leq max_0)$ | (ASSIGN)(11) |
| (13) | $I$ **invof** $exec_{\alpha_1}$ | (ASSIGN)(10)(11)(12) |
| (14) | $at\alpha_2 \wedge I \rightarrow \min(S) = min \wedge max_0 \geq min$ | (prop) |
| (15) | $exec_{\alpha_2} \wedge I \rightarrow \circ(min \leq \min(S))$ | (data)(14)(ASSIGN) |
| (16) | $exec_{\alpha_2} \rightarrow \neg\circ at\{\alpha_1, \alpha_2\}$ | (C1)(C3) |
| (17) | $I$ **invof** $exec_{\alpha_2}$ | (15)(16)(ASSIGN) |
| (18) | $I$ **invof** $\{exec_{\beta_0}, exec_{\beta_1}, exec_{\beta_2}\}$ | (Voraussetzung) |
| (19) | $I$ **invof** $\mathcal{A}_\Pi$ | (prop)(9)(13)(17)(18) |
| (20) | $\Box I$ | (invstart$_\Pi$)(4)(19) |
| (21) | $start_\Pi \rightarrow \Box I$ | (prop)(20) |

b) Beweisen Sie $\mathcal{A}_\Pi \vdash \Box(\exists x \exists y(x \in S \wedge y \in T \wedge x < y) \rightarrow min < max)$.

**Lösung:**

| | | |
|---|---|---|
| (1) | $\Box I$ | (Aufg. a)(root$_\Pi$)(prop) |
| (2) | $\exists x \exists y(x \in S \wedge y \in T \wedge x < y) \rightarrow (\min(S) < \max(T))$ | (data) |
| (3) | $I \rightarrow (min \leq \min(S)) \wedge (\max(T) \leq max)$ | (prop) |
| (4) | $I \wedge \exists x \exists y(x \in S \wedge y \in T \wedge x < y) \rightarrow (min < max)$ | (data)(2)(3) |
| (5) | $\exists x \exists y(x \in S \wedge y \in T \wedge x < y) \rightarrow (min < max)$ | (prop)(1)(ltl3)(4) |
| (6) | $\Box(\exists x \exists y(x \in S \wedge y \in T \wedge x < y) \rightarrow min < max)$ | (alw)(5) |

**Abgabe:** Mittwoch, den 17.1.2007, vor der Übung.