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

WS 2006/2007
Übungsblatt 12

Temporale Logik und Zustandssysteme Lösungsvorschlag

Aufgabe 12-1 **Allgemeine Aussagen über PAR-Programme** (5 Punkte)

Für einen Prozess Π_i eines PAR-Programms Π bezeichne $\mathcal{M}_i$ die Markenmenge von Π_i. Geben Sie die informelle Bedeutung und Herleitungen für folgende Aussagen an:

(at-$\mathcal{M}_i$)	$at\,\mathcal{M}_i$	für jeden Prozess Π_i.
(disjoint)	$exec\lambda \rightarrow (at\,L \leftrightarrow \circ at\,L)$	falls $\lambda \in \mathcal{M}_i$, $L \subseteq \mathcal{M}_j$, $i \neq j$.

Dabei gelte $at\,L \equiv \bigvee_{l \in L} at\,l$ für $L \subseteq \mathcal{M}_i$.

Lösung:

(at-$\mathcal{M}_i$): „Die i-te Komponente wird nie verlassen."

Es sei $\mathcal{A}_{\Pi_i} = \{\lambda_0^{(i)}, \ldots, \lambda_n^{(i)}\}$. (NB: die Syntax von PAR garantiert, dass $\mathcal{A}_{\Pi_i}$ nichtleer und endlich ist.) Je nach Art der Anweisung, die mit $\lambda_j^{(i)}$ markiert ist, trifft Axiom (C1) oder (C2) zu, und wir können folgern $exec\lambda_j^{(i)} \rightarrow \circ at\,\mathcal{A}_{\Pi_i}$, da keine „Sprünge in andere Komponenten" erfolgen.

(1)	$\mathbf{start}_\Pi \rightarrow at\,\lambda_0^{(i)}$	(taut)
(2)	$at\,\lambda_0^{(i)} \rightarrow at\,\mathcal{A}_{\Pi_i}$	(taut)
(3)	$exec\lambda_j^{(i)} \rightarrow \circ at\,\mathcal{A}_{\Pi_i} \quad (j = 0, \ldots, n)$	(C1) bzw. (C2)
(4)	$at\,\lambda_j^{(i)} \wedge \neg exec\lambda_j^{(i)} \rightarrow \circ at\,\lambda_j^{(i)} \quad (j = 0, \ldots, n)$	(C3)
(5)	$\circ at\,\lambda_j^{(i)} \rightarrow \circ at\,\mathcal{A}_{\Pi_i} \quad (j = 0, \ldots, n)$	(taut)(T25)
(6)	$at\,\lambda_j^{(i)} \rightarrow \circ at\,\mathcal{A}_{\Pi_i} \quad (j = 0, \ldots, n)$	(4)(5)
(7)	$at\,\mathcal{A}_{\Pi_i} \rightarrow \bigvee_{j=0}^{n} at\,\lambda_j^{(i)}$	(taut)
(8)	$at\,\mathcal{A}_{\Pi_i} \rightarrow \circ at\,\mathcal{A}_{\Pi_i}$	(6)(7)
(9)	$at\,\mathcal{A}_{\Pi_i} \rightarrow \Box at\,\mathcal{A}_{\Pi_i}$	(ind1)(7)
(10)	$\mathbf{init} \rightarrow \Box at\,\mathcal{A}_{\Pi_i}$	(1)(2)(9)(root)
(11)	$at\,\mathcal{A}_{\Pi_i}$	(init)(10)

(disjoint): „Anweisungen einer Komponente beeinflussen nicht den Kontrollfluss in anderen Komponenten."

(1)	$exec\lambda \rightarrow \neg exec\mu \quad$ (für alle $\mu \in L$)	(I)
(2)	$at\,\mu \wedge \neg exec\mu \rightarrow \circ at\,\mu \quad$ (für alle $\mu \in L$)	(C3)
(3)	$exec\lambda \rightarrow (at\,L \rightarrow \circ at\,L)$	(1)(2)
(4)	$exec\lambda \rightarrow (at(\mathcal{A}_{\Pi_j} \setminus L) \rightarrow \circ at(\mathcal{A}_{\Pi_j} \setminus L))$	(genauso)
(5)	$at\,\mathcal{A}_{\Pi_j}$	(at-Π_j)
(6)	$at(\mathcal{A}_{\Pi_j} \setminus L) \leftrightarrow \neg at\,L$	(5)
(7)	$exec\lambda \rightarrow (\neg at\,L \rightarrow \circ \neg at\,L)$	(4)(6)
(8)	$exec\lambda \rightarrow (at\,L \leftrightarrow \circ at\,L)$	(3)(7)(ltl1)

Aufgabe 12-2 **PAR-Programme: Aushungerungsfreiheit** (9 Punkte)

Gegeben sei das folgende PAR-Programm:

$$
\begin{array}{lll}
\Pi \equiv & \textbf{var } a, b, c : NAT & \\
& \textbf{start } a \geq 0 \wedge b \geq 0 \wedge c \geq 0 & \\
& \textbf{cobegin} & \\
& \quad \textbf{loop} \quad \alpha_0 : c := b; & \textbf{loop} \quad \beta_0 : a := b; \\
& \qquad\quad\ \ \alpha_1 : \textbf{await } a = b; \quad \| \quad\ \ \beta_1 : \textbf{await } b = c; \\
& \qquad\quad\ \ \alpha_2 : a := 0 & \qquad\quad\ \ \beta_2 : c := 1 \\
& \quad \textbf{end} & \textbf{end} \\
& \textbf{coend} &
\end{array}
$$

Es gilt (dies muss *nicht* bewiesen werden):

$$(*) \quad \Diamond(b = c)$$

Weisen Sie die Aushungerungsfreiheit von Π nach, d.h. leiten Sie

$$\Box\Diamond at\alpha_0$$

aus der Menge $\mathcal{A}_\Pi$ der Programmaxiome von Π her.

Dabei dürfen Sie die in Aufgabe 12-1 bewiesenen Aussagen und die abgeleitete Regel

$$
\begin{array}{lll}
\text{(event)} & A \textbf{ invof } Act_\Gamma \setminus Act_h & (\text{mit } Act_h = \{\lambda_1, \ldots, \lambda_m\} \subseteq Act_\Gamma) \\
& exec\lambda \wedge A \rightarrow \circ B & \text{für alle } \lambda \in Act_h \\
& \Box A \rightarrow \Diamond(\text{enabled}_{\lambda_1} \vee \ldots \vee \text{enabled}_{\lambda_m}) & \\
& \vdash A \rightarrow \Diamond B &
\end{array}
$$

ohne Beweis verwenden. Ferner dürfen die Gesetze (T1)–(T31) und die Regel (chain) in der Herleitung verwendet werden.

Lösung: Die Beweisidee besteht darin, die Aussage

$$at\mathcal{M}_0 \rightarrow \Diamond at\alpha_0$$

(mit $\mathcal{M}_0 = \{\alpha_0, \alpha_1, \alpha_2\}$) herzuleiten. Diesen Beweis zerlegen wir in die Teilbehauptungen

$$
\begin{array}{ll}
(1) & at\alpha_0 \rightarrow \Diamond at\alpha_0 \\
(2) & at\alpha_2 \rightarrow \Diamond at\alpha_0 \\
(3) & at\alpha_1 \rightarrow \Diamond at\alpha_2
\end{array}
$$

Nur die Herleitung von Aussage (3) ist nicht unmittelbar klar; die Regel (event) mit hilfreicher Aktion α_1 erfordert den Beweis von

$$\Box at\alpha_1 \rightarrow \Diamond(a = b)$$

Dazu beweisen wir wiederum die Aussage

$$(A) \quad \Diamond(a = b)$$

durch Herleitung von

$$at\mathcal{M}_1 \rightarrow \Diamond(a = b)$$

(mit $\mathcal{M}_1 = \{\beta_0, \beta_1, \beta_2\}$); dieser Beweis ist ähnlich strukturiert wie der der Hauptaussage.

(A.1)	$at\beta_0$ **invof** $\{\alpha_0, \alpha_1, \alpha_2\}$	(prop)(disjoint)
(A.2)	$at\beta_0 \rightarrow \neg at\beta_1$	(PC)
(A.3)	$at\beta_0 \rightarrow \neg exec\beta_1$	(A.2)(action)
(A.4)	$at\beta_0$ **invof** β_1	(A.3)
(A.5)	$at\beta_0$ **invof** β_2	(genauso)

(A.6)	$exec\beta_0 \rightarrow \circ(a = b)$	(assign)
(A.7)	$\Box at\beta_0 \rightarrow \Diamond \text{enabled}_{\beta_0}$	(T4)(T5)
(A.8)	$at\beta_0 \rightarrow \Diamond(a = b)$	(event)(A.1)–(A.7)
(A.9)	$at\beta_1$ **invof** $\{\alpha_0, \alpha_1, \alpha_2\}$	(prop)(disjoint)
(A.10)	$at\beta_1$ **invof** $\{\beta_0, \beta_2\}$	(wie A.4, A.5)
(A.11)	$exec\beta_1 \rightarrow \circ at\beta_2$	(C1)
(A.12)	$\Box at\beta_1 \rightarrow \Diamond \text{enabled}_{\beta_1}$	(*)
(A.13)	$at\beta_1 \rightarrow \Diamond at\beta_2$	(event)(A.9)–(A.12)
(A.14)	$at\beta_2$ **invof** $\{\alpha_0, \alpha_1, \alpha_2\}$	(prop)(disjoint)
(A.15)	$at\beta_2$ **invof** $\{\beta_0, \beta_1\}$	(wie A.4, A.5)
(A.16)	$exec\beta_2 \rightarrow \circ at\beta_0$	(C1)
(A.17)	$\Box at\beta_2 \rightarrow \Diamond \text{enabled}_{\beta_2}$	(T4)(T5)
(A.18)	$at\beta_2 \rightarrow \Diamond at\beta_0$	(event)(A.14)–(A.17)
(A.19)	$at\beta_2 \rightarrow \Diamond(a = b)$	(chain)(A.18)(A.8)
(A.20)	$at\beta_1 \rightarrow \Diamond(a = b)$	(chain)(A.19)(A.13)
(A.21)	$at\mathcal{M}_1 \rightarrow \Diamond(a = b)$	(A.8)(A.19)(A.20)
(A.22)	$at\mathcal{M}_1$	(at-$\mathcal{M}_1$)
(A.23)	$\Diamond(a = b)$	(mp)(A.22)(A.21)

Der Beweis der Behauptungen (1)–(3) folgt nun demselben Schema:

(1)	$at\alpha_0 \rightarrow \Diamond at\alpha_0$	(T5)
(2.1)	$at\alpha_2$ **invof** $\{\beta_0, \beta_1, \beta_2\}$	(prop)(disjoint)
(2.2)	$at\alpha_2$ **invof** $\{\alpha_0, \alpha_1\}$	(wie A.4, A.5)
(2.3)	$exec\alpha_2 \rightarrow \circ at\alpha_0$	(C1)
(2.4)	$\Box at\alpha_2 \rightarrow \Diamond \text{enabled}_{\alpha_2}$	(T4)(T5)
(2.5)	$at\alpha_2 \rightarrow \Diamond at\alpha_0$	(event)(2.1)–(2.4)
(3.1)	$at\alpha_1$ **invof** $\{\beta_0, \beta_1, \beta_2\}$	(prop)(disjoint)
(3.2)	$at\alpha_1$ **invof** $\{\alpha_0, \alpha_2\}$	(wie A.4, A.5)
(3.3)	$exec\alpha_1 \rightarrow \circ at\alpha_2$	(C1)
(3.4)	$\Box at\alpha_1 \rightarrow \Diamond \text{enabled}_{\alpha_1}$	(A)(T29)
(3.5)	$at\alpha_1 \rightarrow \Diamond at\alpha_2$	(event)(3.1)–(3.4)

Nun noch die Herleitung der eigentlichen Behauptung:

(4)	$at\alpha_1 \rightarrow \Diamond at\alpha_0$	(chain)(3)(2)
(5)	$at\mathcal{M}_0 \rightarrow \Diamond at\alpha_0$	(prop)(1)(2)(4)
(6)	$at\mathcal{M}_0$	(at-$\mathcal{M}_0$)
(7)	$\Diamond at\alpha_0$	(mp)(6)(5)
(8)	$\Box\Diamond at\alpha_0$	(alw)(7)

Aufgabe 12-3 **PAR-Programm über Mengen** (keine Abgabe)

Das folgende PAR-Programm wurde bereits in Aufgabe 11-3 betrachtet:

$\Pi \equiv$ **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 $\alpha_0 : min := \min(S)$;
α_1 : **await** $min < max \wedge max < max_0$ **then** $max_0 := max$;
$\alpha_2 : S := (S \setminus \{min\}) \cup \{max_0\}$
end
$\|$
loop $\beta_0 : max := \max(T)$;
β_1 : **await** $max > min \wedge min > min_0$ **then** $min_0 := min$;
$\beta_2 : T := (T \setminus \{max\}) \cup \{min_0\}$
end
coend

Es gilt (ohne Beweis), dass

$$\mathcal{A}_\Pi \vdash \Box\Diamond(at\,\beta_1 \wedge max < max_0).$$

Beweisen Sie:

$$\mathcal{A}_\Pi \vdash \Diamond\forall x \forall y(x \in S \wedge y \in T \rightarrow x \geq y).$$

Lösung: Die Beweisidee ist, vermittels der Regel (fairwfr$_\Pi$) die Lebendigkeitseigenschaft „irgendwann ist jedes Element in S kleiner als jedes Element in T“ nachzuweisen. Dafür benötigen wir eine fundierte Relation $\prec$, so daß $a \prec b$ gdw. a ist „näher an der Lösung“ als b. Dies ist dann gegeben, wenn in a weniger Elemente falsch einsortiert sind. Wir setzen dafür

$$X \equiv \{x \in S \mid \exists y(y \in T \wedge x < y)\}$$

und zeigen $\mathcal{A}_\Pi \vdash \Diamond(X = \emptyset)$, woraus die Behauptung unmittelbar folgt.

Wir stellen fest, daß nicht jede Aktion uns dabei dem „Ziel“ näherbringt. Da wir nur die Menge S betrachten, helfen uns die $exec\,\beta_i$ nicht weiter. Nur $exec\,\alpha_2$ bringt uns bzgl. der Relation weiter (dies aber jedesmal, da S immer geeignet manipuliert wird und insofern X jedesmal kleiner wird). Insofern ist ebenfalls hilfreich, uns auf $exec\,\alpha_2$ „zuzubewegen“.

Wir definieren uns eine fundierte Relation auf Paaren (M, z) aus einer Menge M natürlicher Zahlen und einer Zahl $z \in \{0, 1, 2\}$ wie folgt:

$$(M, z) \prec (M', z') \text{ gdw. } M \subsetneq M' \text{ oder } (M = M' \text{ und } z < z')$$

Die Relation $\prec$ ist, da lexikographisches Produkt zweier fundierter Relationen, selbst fundierte Relation.

Zur Anwendung von (fairwfr$_\Pi$) definieren wir die Formel

$$A((M, z)) \equiv M = X \wedge M \neq \emptyset \wedge at\,\alpha_{2-z}$$

und zeigen $\mathcal{A}_\Pi \exists M \exists z A((M, z)) \rightarrow \Diamond(X = \emptyset)$. Dabei definieren wird die Formel H_λ als

$$H_\lambda \equiv \begin{cases} z = 2 - i & \text{, falls } \lambda \equiv \alpha_i \\ \mathsf{ff} & \text{, falls } \lambda \equiv \beta_i \end{cases}$$

Zuerst wird sichergestellt, daß die $exec\,\beta_i$ unserer fundierten Relation zumindest nicht zuwiderlaufen. Dabei ist insbesondere zu prüfen, daß X durch $exec\,\beta_2$ keine Elemente hinzugewinnt:

(1) I (Teilaufg. a)

(2) $exec\beta_0 \wedge A((M,z)) \rightarrow \circ A((M,z))$ (assign)

(3) $exec\beta_1 \wedge A((M,z)) \rightarrow \circ A((M,z))$ (assign)

(4) $at\beta_2 \wedge I \rightarrow min_0 \leq \min(S)$ (data)

(5) $\{x \in S \mid \exists y(y \in T \setminus \{max\} \wedge x < y)\} \subseteq X$ (data)

(6) $at\beta_2 \wedge I \rightarrow \{x \in S \mid \exists y(y \in (T \setminus \{max\}) \cup \{min_0\} \wedge x < y)\} \subseteq X$ (data)(4)(5)

(7) $exec\beta_2 \rightarrow at\beta_2$ (action)

(8) $exec\beta_2 \wedge A((M,z)) \rightarrow \circ(X = \emptyset \vee \exists M'(M' \subseteq M \wedge A((M',z))))$ (assign)(6)(7)(prop)

(9) $A((M,z)) \rightarrow \neg exec\alpha_i$ für $i \neq 2 - z$ (action)(PC)(prop)

(10) $exec\beta \wedge A((M,z)) \wedge \neg H_\beta \rightarrow \circ(X = \emptyset \vee \exists M' \exists z'((M',z') \preceq (M,z) \wedge A((M',z'))))$(prop)(1)(2)(3)(8)(9)

Dann wird hergeleitet, daß die $exec\alpha_i$ in der Tat eine „Annäherung an das Ziel“ bewirken. Hier ist besonders zu zeigen, daß $exec\alpha_2$ aus X immer Elemente entfernt, also eine echte Teilmengenbeziehung zwischen dem neuen und alten X vorliegt:

(11) $exec\alpha_0 \wedge A((M,z)) \rightarrow z = 2$ (prop)

(12) $exec\alpha_0 \wedge A((M,2)) \rightarrow \circ A((M,1))$ (assign)(C1)(prop)

(13) $exec\alpha_0 \wedge A((M,z)) \rightarrow \circ(X = \emptyset \vee \exists M' \exists z'((M',z') \prec (M,z) \wedge A((M',z'))))$ (12)(prop)

(14) $exec\alpha_1 \wedge A((M,z)) \rightarrow z = 1$ (prop)

(15) $exec\alpha_1 \wedge A((M,1)) \rightarrow \circ A((M,0))$ (assign)(C1)(prop)

(16) $exec\alpha_1 \wedge A((M,z)) \rightarrow \circ(X = \emptyset \vee \exists M' \exists z'((M',z') \prec (M,z) \wedge A((M',z'))))$ (15)(prop)

(17) $at\alpha_2 \wedge I \rightarrow min = \min(S) \wedge max_0 \geq \max(T)$ (data)

(18) $at\alpha_2 \wedge I \wedge X \neq \emptyset \rightarrow min \in X \wedge \neg\exists y(y \in T \wedge max_0 < y)$ (data)(17)

(19) $at\alpha_2 \wedge I \wedge X \neq \emptyset \rightarrow \{x \in (S \setminus \{min\}) \cup \{max_0\} \mid \exists y(y \in T \wedge x < y)\} \subsetneq X$ (data)(18)

(20) $exec\alpha_2 \rightarrow at\alpha_2$ (action)

(21) $exec\alpha_2 \wedge I \wedge A((M,z)) \rightarrow \circ(X = \emptyset \vee \exists M' \exists z'((M',z') \prec (M,z) \wedge A((M',z'))))$ (assign)(data)(19)(20)

(22) $exec\alpha \wedge A((M,z)) \wedge H_\alpha \rightarrow \circ(X = \emptyset \vee \exists M' \exists z'((M',z') \prec (M,z) \wedge A((M',z'))))$(prop)(13)(16)(21)

Drittens ist zu zeigen, daß immer wieder eine hilfreiche Aktion ausführbar ist. Für H_{α_0} und H_{α_2} ist die Ausführbarkeit einfach zu sehen, für H_{α_1} hingegen muß temporallogisch argumentiert werden, daß irgendwann die Bedingung $min < max \wedge max < max_0$ erfüllt ist und insofern kein Deadlock stattfindent:

(23) $A((M,2)) \rightarrow (H_{\alpha_0} \wedge \text{enabled}_{\alpha_0})$ (prop)

(24) $A((M,0)) \rightarrow (H_{\alpha_2} \wedge \text{enabled}_{\alpha_2})$ (prop)

(25) $\Box\Diamond(max < max_0)$ (Vorauss.)(ltl3)(T22)(alw)(prop)

(26) $\Box A((M,1)) \rightarrow \Box(at\alpha_1 \wedge \exists x \exists y(x \in S \wedge y \in T \wedge x < y))$ (data)(T19)(prop)

(27) $\Box A((M,1)) \rightarrow \Box(at\alpha_1 \wedge min < max)$ (26)(11-2-b)(T17)(prop)

(28) $\Box A((M,1)) \rightarrow \Box\Diamond(at\alpha_1 \wedge min < max \wedge max < max_0)$ (25)(27)(temp)(prop)

(29) $\Box A((M,1)) \rightarrow \Diamond(H_{\alpha_1} \wedge \text{enabled}_{\alpha_1})$ (28)(T8)(prop)

(30) $\Box A((M,z)) \rightarrow \Diamond(X = \emptyset \vee E_\Pi)$ (23)(24)(29)(T4)(T5)(prop)

Mit den drei Prämissen kann nun (fairwfr$_\Pi$) angewendet, eine Fallunterscheidung bzgl. des Ausgangszustands ($X = \emptyset$ oder $X \neq \emptyset$) durchgeführt und die Lebendigkeitseigenschaft gefolgert werden:

(31) $\exists M \exists z A((M,z)) \rightarrow \Diamond(X = \emptyset)$ (fairwfr$_\Pi$)(10)(22)(30)

(32) $X \neq \emptyset \rightarrow \exists M \exists z A((M,z))$ (data)

(33) $X = \emptyset \rightarrow \Diamond(X = \emptyset)$ (T5)(prop)

(34) $\Diamond(X = \emptyset)$ (prop)(31)(32)(33)

(35) $X = \emptyset \rightarrow \forall x \forall y(x \in S \wedge y \in T \rightarrow x \geq y)$ (data)

(36) $\Diamond\forall x \forall y(x \in S \wedge y \in T \rightarrow x \geq y)$ (alw)(35)(T26)(prop)(mp)(34)