Reaktive Systeme

Prof. Martin Wirsing

13.12.2002

Transitionssysteme

Reaktive Systeme

Bisher standen folgende Aspekte der Beschreibung von Systemen im Vordergrund:

[Datenstrukturen und Funktionen:] Operationen auf Datenstrukturen durch Angabe des Ein-/Ausgabeverhaltens

[zustandsbasierte (interaktive) Systeme:] Wirkungen von Operationen auf Systemzustand

Reaktive Systeme

- kontinuierliche Interaktion mit Umgebung
- Terminierung unerwünscht bzw. unwesentlich

Beispiele: Prozesssteuerungen, eingebettete Systeme, Protokolle

Formal werden reaktive Systeme durch Transitionssysteme modelliert.
Dabei interessiert sich man meist mehr für den Kontrollfluss als für den Berechnungsaspekt.

Begriff und Beispiele

Wir wiederholen bzw. erweitern die Definition von Transitionssystemen aus Teil III der Vorlesung.

Definition:

Ein markiertes Transitionssystem $\Gamma = (Z, I, \mathcal{A}, \delta)$ mit Anfangszuständen ist gegeben durch

- eine Menge Z von Zuständen,
- eine nichtleere Menge $I \subseteq Z$ von Anfangszuständen,
- eine Menge $\mathcal{A}$ von Aktionen (bzw. Aktionsnamen) und
- eine Zustandsübergangsrelation $\delta \subseteq Z \times \mathcal{A} \times Z$.

Die Aktion A heißt ausführbar ("enabled") im Zustand s, wenn ein t existiert mit $(s, A, t) \in \delta$.
Im folgenden setzen wir voraus, dass in jedem Zustand $s \in Z$ mindestens eine Aktion $A \in \mathcal{A}$ ausführbar ist (d.h. δ ist total).

Ein Ablauf von Γ ist eine unendliche Folge $\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$, so dass gilt:

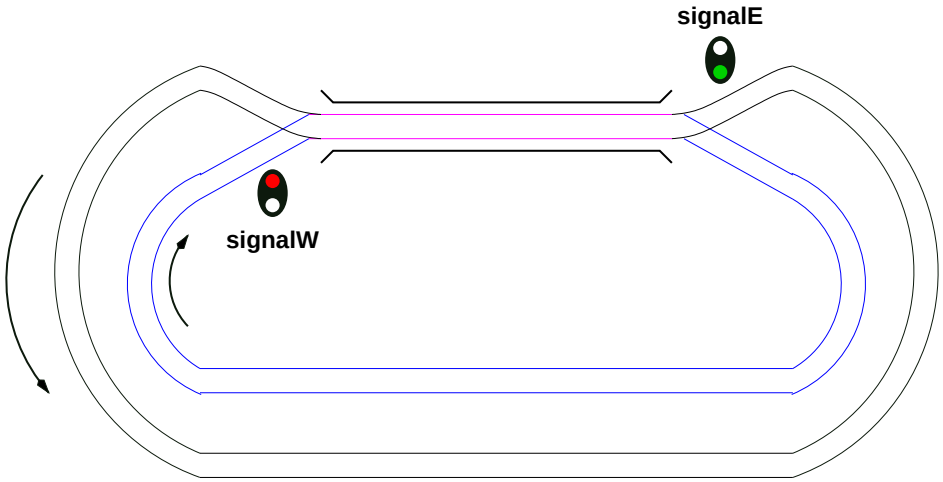
- $s_0 \in I$ ist ein Anfangszustand von Γ und
- für alle $i \geq 0$ ist $(s_i, A_i, s_{i+1}) \in \delta$.

Ein endlicher Ablauf von Γ ist ein endlicher Präfix eines Ablaufs von Γ.

Bemerkungen

- Transitionssysteme haben eine ähnliche Struktur wie endliche Automaten, aber keine Endzustände.
- Definitionen in der Literatur uneinheitlich, z.B.:
  - ein Anfangszustand $s_0 \in Z$ statt Menge $I \subseteq Z$
  - Aktionen oft implizit gelassen
  - Totalität von δ nicht gefordert, gelegentlich auch endliche Abläufe erlaubt
  - explizite "Stotteraktion" τ mit $(s, \tau, t) \in \delta$ gdw. $s = t$
- Die Zustandsmenge Z darf unendlich (sogar überabzählbar) sein.
- Zustandsübergänge modelliert als "atomare" Aktionen ohne zeitliche Dauer.
- Die Zustandsmenge Z wird meist durch Belegungen einer Menge $\mathcal{V}$ von (Zustands-)Variablen angegeben, evtl. eingeschränkt durch eine Zustandsinvariante.
- Jede Z-Spezifikation der Form $\langle State, Init, Ops \rangle$ definiert ein Transitionssystem mit Aktionenmenge Ops
  (vorausgesetzt, in jedem Zustand ist mindestens eine Operation ausführbar).

Example (Eisenbahnsteuerung als Transitionssystem):



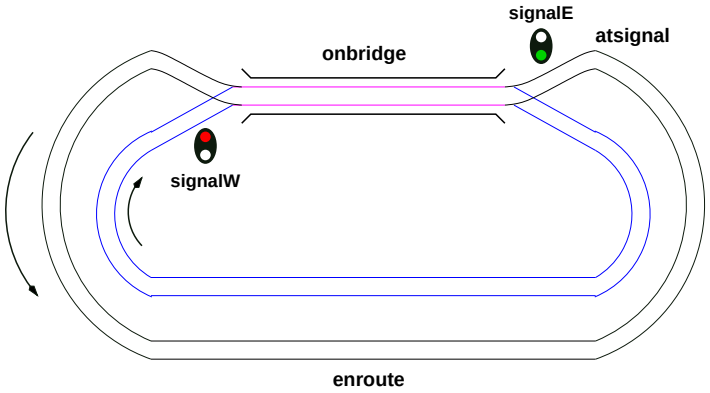
Zustand der Steuerung beinhaltet Informationen über:

- Signalstellung
- Position, Geschwindigkeit und Beschleunigung der Züge

Zustandsübergänge:

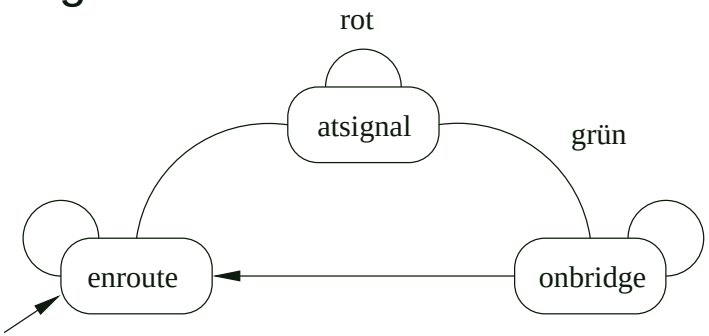
- Erfassen neuer Zugdaten durch Sensoren
- Schalten der Signale

Abstraktere Modellierung führt zu endlichem Zustandsraum, z.B. Einteilung der Strecke in Abschnitte



Zustandsübergänge

Züge

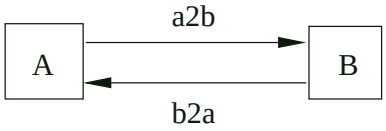


Nichtdeterminismus wegen abstrakter Modellierung

Signale (kombinatorische Schaltung)

trainW	trainE	signalW	signalE
enroute	atsignal	rot	grün
atsignal	enroute	grün	rot
atsignal	atsignal	?	?
sonst		rot	rot

Example (Alternating-Bit-Protokoll):



- Übertragung von Nachrichten über unzuverlässigen Kanal
- Kanal kann Nachrichten verlieren (aber nicht umordnen oder verfälschen)
- Sender verschickt Sequenzbits
- Empfänger bestätigt erhaltene Bits (über unzuverlässigen Rückkanal)

Protokoll

- Sender versendet Nachricht (msg, b) so lange, bis Bestätigung für aktuelles Sequenzbit vom Empfänger eintrifft (also $ack = b$ gilt).
- Für die folgende Nachricht verwendet der Sender das Sequenzbit $1 - b$.
- Empfänger schickt erhaltenes Sequenzbit an Sender zurück.
- Hat das Sequenzbit gewechselt, gibt der Empfänger die Nachricht aus.

Modellierung als Transitionssystem (in Z-Notation)

1. Zustände gegeben durch die folgenden Zustands-Schemata

$[MSG]$ Basistyp der zu versendenden Nachrichten

$BIT == 0 \,..\, 1$

Sender

in : seq MSG

$curr : \mathbb{N}_1$

$b : BIT$

Channel

$msgCh$: seq$(MSG \times BIT)$

$ackCh$: seq BIT

Receiver

$lastb : BIT$

out : seq MSG

ABP

$Sender$

$Channel$

$Receiver$

InitABP

ABP

$b \neq lastb$

$curr = 1$

$msgCh = ackCh = \langle\rangle$

$out = \langle\rangle$

2. Aktionen beschrieben durch die folgenden Operations-Schemata

SendMsg

$\Delta ABP, \Xi Sender, \Xi Receiver$

$curr \in \mathrm{dom}\, in$
$msgCh' = msgCh \frown \langle (in\, curr, b) \rangle$
$ackCh' = ackCh$

EnvSend

$\Delta ABP, \Xi Channel, \Xi Receiver$
$m? : MSG$

$in' = in \frown \langle m? \rangle$
$curr' = curr$
$b' = b$

Lose

$\Delta ABP, \Xi Sender, \Xi Receiver$

$\exists I, J : \mathbb{N} \bullet$
$msgCh' = msgCh \upharpoonright I$
$\wedge\ ackCh' = ackCh \upharpoonright J$

RcvMsg

$\Delta ABP, \Xi Sender$

$msgCh \neq \langle\rangle$
$msgCh' = tail\, msgCh$
$lastb' = (head\, msgCh).2$
$out' = \textbf{if}\ lastb' \neq lastb$
$\qquad \textbf{then}\ out \frown \langle (head\, msgCh).1 \rangle\ \textbf{else}\ out$
$ackCh' = ackCh \frown \langle lastb' \rangle$

RcvAck

$\Delta ABP, \Xi Receiver$

$ackCh \neq \langle\rangle$
$ackCh' = tail\, ackCh$
$in' = in$
$curr' = \textbf{if}\ head\, ackCh = b\ \textbf{then}\ curr + 1\ \textbf{else}\ curr$
$b' = \textbf{if}\ head\, ackCh = b\ \textbf{then}\ 1 - b\ \textbf{else}\ b$
$msgCh' = msgCh$

Die Spezifikation enthält drei Komponenten: Sender, Empfänger und Kanal.

- Die Umgebungsaktion $EnvSend$ bringt eine neue Nachricht ins System ein.
- Die Aktionen $SendMsg$, $RcvMsg$ und $RcvAck$ modellieren das eigentliche Protokoll.
- Die Aktion $Lose$ modelliert den Verlust beliebig vieler Nachrichten auf dem Übertragungskanal.
- Die Aktionen $EnvSend$ und $Lose$ sind immer ausführbar.

Die Spezifikation erlaubt u.a. folgende Arten von Abläufen:

- $EnvSend$ wird nur endlich oft ausgeführt, der Kanal ist ab einem gewissen Zeitpunkt immer leer, und es finden nur noch $Lose$-Aktionen statt: “Terminierung” (quiescence).
- Alle Nachrichten (oder Bestätigungen) gehen verloren, und das Protokoll macht keinen Fortschritt (livelock).

Interessante Eigenschaften:

- Kommt jede gesendete Nachricht irgendwann beim Empfänger an?
- Bleibt die Reihenfolge der Nachrichten erhalten?

Programme als Transitionssysteme

Example (Stoppuhr):

var x, y : integer = $0, 0$;
cobegin
α : **while** $y = 0$ **do** $\beta : x := x + 1$ **end** $\quad \| \quad \gamma : y := 1$
coend

Zustände: Belegungen der Programmvariablen, außerdem “Programmzähler”

Aktionen: entsprechend Programmanweisungen, “Stottern” am Programmende

mögliche Abläufe

Ablauf 1:

Aktion	—	α	β	α	β	α	γ	β	α	τ	$\cdots$
x	0	0	1	1	2	2	2	3	3	3	$\cdots$
y	0	0	0	0	0	0	1	1	1	1	$\cdots$

Ablauf 2:

Aktion	—	α	β	α	β	α	β	α	β	α	$\cdots$
x	0	0	1	1	2	2	3	3	4	4	$\cdots$
y	0	0	0	0	0	0	0	0	0	0	$\cdots$

Fairnessbedingungen

Transitionssysteme erlauben häufig “unfaire” Abläufe, die im modellierten System nicht vorkommen können.

Typische Ursachen unfairer Abläufe:

- Abstraktion bei Modellierung führt zu Nichtdeterminismus im Modell
  vgl. Beispiel Eisenbahn: Transitionssystem erlaubt Zyklen, in denen ein Zug immer auf der Brücke bleibt.

- Modellierung von Parallelität durch Nichtdeterminismus
  vgl. Beispiel Stoppuhr: In Ablauf 2 wird Aktion γ nie ausgeführt.

- Zwei Aktionen stehen in Konflikt, d.h. sind im selben Zustand ausführbar
  vgl. Beispiel Alternating-Bit-Protokoll: korrekte und fehlerhafte Nachrichtenübertragung ($RcvMsg$ vs. $Lose$)

Fairnessbedingungen schränken die Menge der erlaubten Abläufe ein, indem unfaire Abläufe ausgeschlossen werden.

Typische Fairnessbedingungen.

- Beispiel Eisenbahn: Enthält ein Ablauf unendlich viele Bewegungen eines Zugs ausgehend vom Abschnitt onbridge, so befindet sich der Zug unendlich oft nicht im Abschnitt onbridge.

- Beispiel Alternating-Bit-Protokoll: Werden immer wieder Nachrichten auf den Kanal gesendet, so werden auch immer wieder Nachrichten empfangen.
  Also: Wird $SendMsg$ unendlich oft ausgeführt, dann auch $RcvMsg$.

- Beispiel Stoppuhr: Ein Prozess, der (ab einem gewissen Zeitpunkt) ständig ausführbereit ist, führt irgendwann eine Aktion aus.

Fairnessbedingungen fordern intuitiv, dass Aktionen, die “genügend häufig” ausführbar sind, irgendwann ausgeführt werden. Wir betrachten zwei Fairnessbegriffe:

schwache Fairness (weak fairness, justice): Ein Ablauf $s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$ heißt schwach fair bezüglich der Aktion A, falls gilt:
Existiert ein $n \in \mathbb{N}$, so dass A in allen Zuständen s_m mit $m \geq n$ ausführbar ist, so ist $A_i = A$ für unendlich viele $i \in \mathbb{N}$.
Äquivalente Formulierung: Wird A nur endlich oft ausgeführt, so ist A im Ablauf unendlich oft nicht ausführbar.

starke Fairness (strong fairness, compassion): Ein Ablauf $s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$ heißt stark fair bezüglich der Aktion A, falls gilt:
Existieren unendlich viele $m \in \mathbb{N}$, so dass A im Zustand s_m ausführbar ist,
so ist $A_i = A$ für unendlich viele $i \in \mathbb{N}$.
Äquivalente Formulierung: Wird A nur endlich oft ausgeführt, so ist A im Ablauf nur endlich oft ausführbar.

Bemerkung: Starke Fairness impliziert schwache Fairness.
Ist ein Ablauf stark fair bezüglich A, so ist er auch schwach fair bezüglich A.

Definition:

Ein Transitionssystem mit Fairness (fair transition system, FTS) $\Gamma_f = (Z, I, \mathcal{A}, \delta, W, S)$ erweitert ein Transitionssystem $\Gamma = (Z, I, \mathcal{A}, \delta)$ um Mengen $W, S \subseteq \mathcal{A}$.
Die Abläufe von Γ_f sind diejenigen Abläufe von Γ, die schwach fair sind bezüglich der Aktionen $A \in W$ und stark fair bezüglich der Aktionen $A \in S$.

Für die betrachteten Beispiele sind folgende Fairnessbedingungen sinnvoll:

Beispiel Eisenbahn: schwache Fairness für die Aktionen “Brücke verlassen” mit der Vorbedingung “Zug im Abschnitt onbridge” und der Nachbedingung “Zug im Abschnitt enroute”.

Beispiel Alternating-Bit-Protokoll

- schwache Fairness für die Aktion $SendMsg$
- starke Fairness für die Aktionen $RcvMsg$ und $RcvAck$

Beispiel Stoppuhr: schwache Fairness für jeden der beiden Prozesse, d.h. für die Aktionen $P1$ und $P2$ mit

$$(s, P1, t) \in \delta \quad \Leftrightarrow \quad (s, \alpha, t) \in \delta \text{ oder } (s, \beta, t) \in \delta$$
$$(s, P2, t) \in \delta \quad \Leftrightarrow \quad (s, \gamma, t) \in \delta$$

Die Wahl adäquater Fairnessbedingungen hängt vom jeweils modellierten System ab und ist häufig der schwierigste Teil der Modellierung.

Implementierung von Fairnessbedingungen

Fairnessbedingungen können durch geeignete Scheduler implementiert werden. Für schwache Fairness genügt ein “round-robin-Scheduler”.

Theorem:

Es sei $\Gamma_f = (Z, I, \mathcal{A}, \delta, \{B_0, \ldots, B_{m-1}\}, \emptyset)$ ein FTS ohne starke Fairness, und $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$ sei ein endlicher Ablauf von Γ.

Dann ist jede Folge $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n \xrightarrow{A_n} s_{n+1} \ldots$ ein Ablauf von Γ_f, falls für alle $k \geq n$ die folgenden Bedingungen erfüllt sind:

1. $A_k = B_{k \bmod m}$, falls die Aktion $B_{k \bmod m}$ im Zustand s_k ausführbar ist.
2. $(s_k, A_k, s_{k+1}) \in \delta$.

Bemerkung: Wegen der angenommenen Totalität von δ kann jeder endliche Ablauf von Γ_f zu einem Ablauf erweitert werden, der die Bedingungen von Satz (siehe oben) erfüllt.

Beweis des Satzes

Angenommen, $\sigma = s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n \xrightarrow{A_n} s_{n+1} \ldots$ erfülle die Bedingungen (1) und (2), sei aber nicht schwach fair bezüglich der Aktion B_i.

Da $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$ endlicher Ablauf von Γ ist und σ die Bedingung (2) erfüllt, ist σ sicher ein Ablauf von $\Gamma = (Z, I, \mathcal{A}, \delta)$.

Es sei $p \in \mathbb{N}$ so gewählt, dass B_i in allen Zuständen s_j mit $j \geq p$ ausführbar ist.
Da σ die Fairnessbedingung für B_i verletzt, ist $A_k = B_i$ nur für endlich viele $k \in \mathbb{N}$.

Wähle $q \geq p$ so, dass $q \bmod m = i$ gilt und $A_k \neq B_i$ ist für alle $k \geq q$.

Dann ist B_i ausführbar im Zustand s_q, also folgt nach Bedingung (1), dass $A_q = B_i$ gilt.
Widerspruch.

Die Implementierung von starken Fairnessbedingungen benötigt einen Scheduler mit “dynamischen Prioritäten”: Aktionen werden um so stärker priorisiert, je länger sie nicht ausgeführt wurden.

Theorem:

Es sei $\Gamma_f = (Z, I, \mathcal{A}, \delta, \emptyset, \{B_0, \ldots, B_{m-1}\})$ ein FTS mit starken Fairnessbedingungen, und $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$ sei ein endlicher Ablauf von Γ.

Dann ist jede Folge $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n \xrightarrow{A_n} s_{n+1} \ldots$ ein Ablauf von Γ_f, falls eine Folge $\pi_n, \pi_{n+1}, \ldots$ von Permutationen von $\{0, \ldots, m-1\}$ existiert, so dass für alle $k \geq n$ die folgenden Bedingungen erfüllt sind:

1. Falls $i \in \{0, \ldots, m-1\}$ existiert, so dass B_i im Zustand s_k ausführbar ist:
   Sei $j \in \{0, \ldots, m-1\}$ minimal, so dass $B_{\pi_k(j)}$ in s_k ausführbar ist, dann gilt
   $$A_k = B_{\pi_k(j)} \quad \text{und} \quad \pi_{k+1}(i) = \left\{ \begin{array}{ll} \pi_k(i) & \text{falls } i < j \\ \pi_k(i+1) & \text{falls } j \leq i < m-1 \\ \pi_k(j) & \text{falls } i = m-1 \end{array} \right\} \text{ für alle } i \in \{0, \ldots, m-1\}$$
2. Sonst ist A_k beliebige Aktion, die in s_k ausführbar ist, und $\pi_{k+1} = \pi_k$.

3. $(s_k, A_k, s_{k+1}) \in \delta$.

Bemerkung: Wieder kann jeder endliche Ablauf von Γ zu einem Ablauf erweitert werden, der die Bedingungen dieses Satzes erfüllt. Bedingung (1) besagt intuitiv, dass die gerade ausgeführte Aktion niedrigste Priorität erhält.

Beweis des Satzes

Angenommen, $\sigma = s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n \xrightarrow{A_n} s_{n+1} \ldots$ erfülle die Bedingungen (1), (2) und (3), sei aber kein Ablauf von Γ_f.

Da $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$ endlicher Ablauf von Γ ist und σ die Bedingung (3) erfüllt, ist σ sicher ein Ablauf von $\Gamma = (Z, I, \mathcal{A}, \delta)$.

Angenommen, σ ist nicht stark fair bezüglich der Aktion B_i. Dann gibt es unendlich viele $m \in \mathbb{N}$, so dass B_i in s_m ausführbar ist, aber $A_k = B_i$ gilt nur für endlich viele k.

Sei $p \geq n$ so gewählt, dass $A_k \neq B_i$ gilt für alle $k \geq p$. Betrachte die Folge $\pi_p, \pi_{p+1}, \ldots$. Da $A_k \neq B_i$ gilt, ist die Folge $j_p, j_{p+1}, \ldots$ mit $\pi_k(j_k) = i$ (schwach) monoton abnehmend, wird also schließlich stabil.

Wähle $q \geq p$ so, dass $j_k = j_q$ für alle $k \geq q$. Dann folgt $\pi_k(j) = \pi_q(j)$ für alle $j \leq j_q$ und $k \geq q$, und die Aktionen $B_{\pi_q(1)}, \ldots, B_{\pi_q(j_q)}$ sind in keinem Zustand s_k (für $k \geq q$) ausführbar — denn sonst wäre gemäß Bedingung (1a) $A_k = B_{\pi_k(j)}$ für ein $j \leq j_q$, und damit $j_{k+1} \neq j_k$.

Insbesondere ist $B_{\pi_q(j_q)} = B_i$ in keinem Zustand s_k (für $k \geq q$) ausführbar, Widerspruch.

Interpretation: Die Sätze besagen intuitiv:

- Ist Γ_f ein FTS, dessen zu Grunde liegendes Transitionssystem Γ ohne Fairnessbedingungen ausführbar ist (d.h. die Menge der Nachfolgerzustände zu jeder Aktion ist berechenbar), so können auch faire Abläufe von Γ_f systematisch erzeugt werden.

  Dabei reicht es sogar, den Scheduler erst ab einem beliebigen Zeitpunkt nach Beginn des Ablaufs zu verwenden.
- Allerdings werden auf diese Weise nicht alle fairen Abläufe von Γ_f generiert (selbst endliche FTSe haben i.a. überabzählbar unendlich viele verschiedene faire Abläufe).
- Der prioritätsbasierte Scheduler kann auch für FTSe verwendet werden, die sowohl starke wie schwache Fairnessbedingungen enthalten, da starke Fairness schwache Fairness impliziert.

Eigenschaften von Abläufen

Bei der Analyse von Transitionssystemen interessiert man sich meist für Aussagen über ihre Abläufe, z.B.:

- Die Züge können zu keinem Zeitpunkt beide im Abschnitt onbridge sein.
- Wartet ein Zug vor dem Signal, so wird er zu einem späteren Zeitpunkt auf der Brücke sein.
- Jede empfange Nachricht wurde zuvor vom Sender verschickt.
- Die Reihenfolge der empfangenen Nachrichten entspricht der Reihenfolge beim Senden.
- Jede gesendete Nachricht trifft irgendwann beim Empfänger ein.

Außerdem interessiert man sich gelegentlich für Aussagen über die Verzweigungsstruktur von Transitionssystemen wie:

- Aktionen A und B stehen in Konflikt oder sind unabhängig.
- Von jedem Zustand aus kann ein Anfangszustand erreicht werden.
- Zwei Prozesse können kooperieren, um einen dritten Prozess auszusperren.

Solche Aussagen klammern wir hier aus (vgl. Vorlesungen Prozessalgebra, temporale Logik).

Wir identifizieren Eigenschaften mit der Menge der Zustands-Aktions-Folgen, welche die Eigenschaft erfüllen.

Definition:

Sei $\Sigma = (Z, \mathcal{A})$ eine Menge von Zuständen und Aktionen.

Eine (Σ-)Eigenschaft ist eine Menge von Folgen $\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$ mit $s_i \in Z$ und $A_i \in \mathcal{A}$.

Eine “Eigenschaft” P ist also eine Aussage, von der es sinnvoll ist zu fragen, ob sie auf einen Ablauf σ zutrifft ($\sigma \in P$) oder nicht.

Aussagen über die Existenz von Abläufen sind keine “Eigenschaften” im Sinne dieser Definition.

Beispiele:

- Menge der Abläufe eines Transitionssystems Γ
- Menge der Zustands-Aktions-Folgen, die stark fair sind bzgl. Aktion A
- Menge aller Folgen $s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$, so dass $s_n(y) = 1$ gilt für ein $n \in \mathbb{N}$

Sicherheits- und Lebendigkeitseigenschaften

- grundlegende Klassen von Eigenschaften mit unterschiedlichen Beweisprinzipien
- Verallgemeinerung von partieller Korrektheit und Terminierung bei sequentiellen Programmen

Informelle Charakterisierung (Lamport 1980)

Sicherheitseigenschaft (safety property): *something bad never happens*

- kein Zusammenstoß auf der Brücke
- Reihenfolge der empfangenen Nachrichten entspricht Reihenfolge beim Senden

Lebendigkeitseigenschaft (liveness property): *something good eventually happens*

- Züge können Brücke irgendwann befahren
- Nachrichten werden unendlich oft übertragen
- Aktion γ wird irgendwann ausgeführt

Formale Charakterisierung (Alpern, Schneider 1985)

Definition:

Sei $\Sigma = (Z, \mathcal{A})$ gegeben.

P ist eine (Σ-)Sicherheitseigenschaft genau dann, wenn für jede Folge $\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$ gilt:

$\sigma \in P$ gdw. für jeden Präfix $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$ von σ gibt es eine Folge $\tau = s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n \xrightarrow{B_n} t_{n+1} \xrightarrow{B_{n+1}} t_{n+2} \ldots$ mit $\tau \in P$

P ist eine (Σ-)Lebendigkeitseigenschaft genau dann, wenn sich jede endliche Folge $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$ zu einer Folge $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n \xrightarrow{A_n} s_{n+1} \ldots \in P$ ergänzen lässt.

Zusammenhang zu informeller Charakterisierung

- Eine Folge σ erfüllt eine Sicherheitseigenschaft P genau dann nicht, wenn es einen endlichen Präfix von σ gibt, der sich nicht zu einer Folge ergänzen lässt, die P erfüllt.
- Lebendigkeitseigenschaften schränken dagegen die Menge der endlichen Präfixe nicht ein.

Vereinfachende Schreibweisen ($\Sigma = (Z, \mathcal{A})$ sei vorausgesetzt)

- Für eine Folge $\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$ bezeichne $\sigma[..n]$ den Präfix $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$.
- Für Folgen $\rho = s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$ und $\sigma = s_n \xrightarrow{A_n} s_{n+1} \xrightarrow{A_{n+1}} s_{n+2} \ldots$ bezeichnet $\rho \circ \sigma$ die Konkatenation $s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n \xrightarrow{A_n} s_{n+1} \xrightarrow{A_{n+1}} s_{n+2} \ldots$.
- Für eine endliche Folge $\rho = s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} s_n$ schreiben wir $\rho \in P$, falls $\rho = \sigma[..n]$ gilt für ein $\sigma \in P$ und ein $n \in \mathbb{N}$.
- Σ^* und Σ^ω bezeichnen die Menge aller endlichen bzw. unendlichen Folgen von Zuständen und Aktionen.

Damit gilt:

P ist Sicherheitseigenschaft genau dann, wenn für alle Folgen $\sigma \in \Sigma^\omega$ gilt:

Falls $\sigma[..n] \in P$ für alle $n \in \mathbb{N}$, so gilt auch $\sigma \in P$.

P ist Lebendigkeitseigenschaft genau dann, wenn $\sigma[..n] \in P$ gilt für alle Folgen $\sigma \in \Sigma^\omega$ und alle $n \in \mathbb{N}$.

Beobachtung: Die Menge der Abläufe eines Transitionssystems $\Gamma = (Z, I, \mathcal{A}, \delta)$ ohne Fairnessbedingungen ist eine Sicherheitseigenschaft.

Denn: Sei $\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$

	σ Ablauf von Γ	
gdw.	$s_0 \in I$ und für alle $i \in \mathbb{N}$ gilt $(s_i, s_{i+1}) \in \delta$	[siehe Definition]
gdw.	für alle $n \in \mathbb{N}$ gilt $s_0 \in I$ und für alle $i < n$ gilt $(s_i, s_{i+1}) \in \delta$	[Arithmetik]
gdw.	für alle $n \in \mathbb{N}$ gibt es $\tau \in \Sigma^\omega$, so dass $\sigma[..n] \circ \tau$ Ablauf von Γ	[wegen Totalität von δ]

Dagegen sind Fairnessbedingungen Lebendigkeitseigenschaften:

Wie in den vorangegangenen Sätzen bewiesen, kann jede Folge $\rho \in \Sigma^*$ zu einer Folge $\sigma \in \Sigma^\omega$ ergänzt werden, welche die Fairnesseigenschaft erfüllt.

Example (Sicherheits- und Lebendigkeitseigenschaften: Alternating-Bit-Protokoll):

Die Eigenschaft “Reihenfolge der empfangenen Nachrichten entspricht Reihenfolge beim Senden” (formal: $out \sqsubseteq in$ gilt für alle Zustände eines Ablaufs) ist eine Sicherheitseigenschaft.

Denn trifft sie auf eine Folge $\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$ nicht zu, so gibt es einen (kleinsten) Index $n \in \mathbb{N}$, so dass $out \sqsubseteq in$ im Zustand s_n nicht erfüllt ist. Damit verletzt bereits $s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots \xrightarrow{A_{n-1}} s_n$ die Eigenschaft.

Dagegen ist die Eigenschaft L “die Nachricht “ok” kommt unendlich oft beim Empfänger an” eine Lebendigkeitseigenschaft.

Denn: Seien $\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots \in \Sigma^\omega$ und $n \in \mathbb{N}$ beliebig.

Definiere den Zustand t_i durch $\left\{ \begin{array}{ll} t_i(out) = s_n(out) \frown \langle \underbrace{\text{“ok”}, \ldots, \text{“ok”}}_{i\text{-mal}} \rangle & \\ t_i(x) = s_n(x) & \text{für } x \not\equiv out \end{array} \right\}$ (insbes. ist $t_0 = s_n$)

Dann erfüllt die Folge $\tau = s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{n-1}} t_0 \xrightarrow{A_0} t_1 \xrightarrow{A_0} s \ldots$ die Eigenschaft L, also gilt auch $\sigma[..n] \in L$.

Sicherheits- und Lebendigkeitseigenschaften: Ergebnisse

Theorem:

1. Die Klasse der Sicherheitseigenschaften ist abgeschlossen unter beliebigen Durchschnitten.
2. Ist L eine Lebendigkeitseigenschaft, so ist auch jede Eigenschaft $M \supseteq L$ Lebendigkeitseigenschaft.
3. Für jede Eigenschaft P ist

   $$\mathcal{C}(P) = \{\sigma \in \Sigma^\omega \mid \sigma[..n] \in P \text{ für alle } n \in \mathbb{N}\}$$

   die (bzgl. $\subseteq$) kleinste Sicherheitseigenschaft, die P enthält.
   $\mathcal{C}(P)$ heißt der Sicherheitsabschluss (safety closure) von P.
4. P ist Sicherheitseigenschaft genau dann, wenn $\mathcal{C}(P) = P$ gilt.
5. P ist Lebendigkeitseigenschaft genau dann, wenn $\mathcal{C}(P) = \Sigma^\omega$ gilt.
6. Σ^ω ist die einzige Eigenschaft, die sowohl Sicherheits- als auch Lebendigkeitseigenschaft ist.

7. Jede Eigenschaft P lässt sich als Durchschnitt einer Sicherheits- und einer Lebendigkeitseigenschaft darstellen.
8. Ist S Sicherheitseigenschaft und P beliebige Eigenschaft, so gilt

   $$P \subseteq S \quad \text{gdw.} \quad \mathcal{C}(P) \subseteq S$$

Proof

1. Sei $\mathcal{S}$ eine Menge von Sicherheitseigenschaften, zu zeigen: $\bigcap \mathcal{S}$ ist Sicherheitseigenschaft.
   Sei $\sigma \in \Sigma^\omega$ beliebig mit $\sigma[..n] \in \bigcap \mathcal{S}$ für alle $n \in \mathbb{N}$.
   Also gilt $\sigma[..n] \in S$ für alle $n \in \mathbb{N}$ und alle $S \in \mathcal{S}$.
   Da jedes $S \in \mathcal{S}$ Sicherheitseigenschaft ist, folgt $\sigma \in S$ für alle $S \in \mathcal{S}$.
   Daher gilt $\sigma \in \bigcap \mathcal{S}$, und damit die Behauptung.
2. unmittelbar nach Definition.

3. Offenbar gilt $P \subseteq \mathcal{C}(P)$.

   Ferner ist $\mathcal{C}(P)$ Sicherheitseigenschaft: Sei $\sigma \in \Sigma^\omega$ beliebig mit $\sigma[..n] \in \mathcal{C}(P)$ für alle $n \in \mathbb{N}$.

   Das heißt, für alle $n \in \mathbb{N}$ existiert $\tau_n \in \Sigma^\omega$ mit $\sigma[..n] \circ \tau_n \in \mathcal{C}(P)$.

   Nach Definition von $\mathcal{C}(P)$ folgt $\sigma[..n] \in P$ für alle $n \in \mathbb{N}$, also $\sigma \in \mathcal{C}(P)$.

   Sei nun $S \supseteq P$ beliebige Sicherheitseigenschaft, zu zeigen ist $\mathcal{C}(P) \subseteq S$.

   Sei $\sigma \in \mathcal{C}(P)$ beliebig.

   Dann gilt $\sigma[..n] \in P$ für alle $n \in \mathbb{N}$, und wegen $P \subseteq S$ folgt $\sigma[..n] \in S$ für alle $n \in \mathbb{N}$.

   Da S Sicherheitseigenschaft ist, folgt $\sigma \in S$, was zu zeigen war.

4. Übung.

5. Übung.

6. Σ^ω ist offensichtlich Sicherheits- und Lebendigkeitseigenschaft.

   Umgekehrt sei P Sicherheits- und Lebendigkeitseigenschaft, zu zeigen: $P = \Sigma^\omega$.
   Sei $\sigma \in \Sigma^\omega$ beliebig.
   Da P Lebendigkeitseigenschaft ist, gilt $\sigma[..n] \in P$ für alle $n \in \mathbb{N}$.
   Da P Sicherheitseigenschaft ist, folgt daraus $\sigma \in P$.

7. Sei $L =_{\text{def}} P \cup (\Sigma^\omega \setminus \mathcal{C}(P))$.

   Offensichtlich gilt $P = \mathcal{C}(P) \cap L$.
   Zu zeigen bleibt: L ist Lebendigkeitseigenschaft.
   Sei $\sigma \in \Sigma^\omega$ beliebig, zu zeigen: $\sigma[..n] \in L$ gilt für alle $n \in \mathbb{N}$.
   Fall 1: $\sigma \notin \mathcal{C}(P)$. Dann ist $\sigma \in L$, also offenbar $\sigma[..n] \in L$ für alle $n \in \mathbb{N}$.
   Fall 2: $\sigma \in \mathcal{C}(P)$. Dann gilt für alle $n \in \mathbb{N}$, dass $\sigma[..n] \in P \subseteq L$.

8. Übung.

Example (vgl. Beispiel Stoppuhr):

Sei P die Menge aller Folgen $s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$, für die gilt:
Es existiert ein $n \geq 0$, so dass gilt:

- $s_i(y) = 0$ für alle $i \leq n$ und $s_i(y) = 1$ für alle $i > n$,

Intuitive Bedeutung: Eine Zeit lang gilt $y = 0$, danach $y = 1$.

Der Sicherheitsabschluss $\mathcal{C}(P)$ enthält genau die Folgen $\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$, für die gilt:

- $\sigma \in P$ oder
- $s_i(y) = 0$ für alle $i \in \mathbb{N}$.

Dieser Satz besagt, dass sich jede Spezifikation als Paar (S, L) aus einer Sicherheitseigenschaft S und einer Lebendigkeitseigenschaft L schreiben lässt.

Eine natürliche Forderung besagt, dass L die gemäß S erlaubten Teilabläufe nicht einschränkt.

Definition:

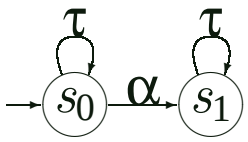
Sei S Sicherheitseigenschaft und L beliebige Eigenschaft.

Das Paar (S, L) heißt maschinenabgeschlossen (machine closed), wenn $\mathcal{C}(S \cap L) = S$ gilt.

Interpretation: Jede endliche Folge $\rho \in S$ lässt sich zu einer Folge $\rho \circ \tau \in S \cap L$ ergänzen.

Example (nicht maschinenabgeschlossene Spezifikation):

Sei S die Menge aller Abläufe des folgenden Transitionssystems



und L die Menge aller Folgen, die unendlich oft den Zustand s_0 enthalten.

Der endliche Ablauf $s_0 \xrightarrow{\alpha} s_1$ lässt sich nicht zu einem Ablauf in $S \cap L$ ergänzen.

Die Abläufe einer maschinenabgeschlossenen Spezifikation (S, L) erfüllen eine Sicherheitseigenschaft I genau dann, wenn $S \subseteq I$ gilt:

	$S \cap L \subseteq I$	
gdw.	$\mathcal{C}(S \cap L) \subseteq I$	[siehe Satz]
gdw.	$S \subseteq I$	[(S, L) maschinenabgeschlossen]

Die zusätzliche Eigenschaft L wird also zum Beweis von Sicherheitseigenschaften nicht benötigt.

Nicht maschinenabgeschlossene Spezifikationen sind problematisch und werden in einigen Formalismen ganz ausgeschlossen.

Aus den Sätzen WF-Scheduler und SF-Scheduler folgt: FTSe mit endlich vielen Fairnessbedingungen sind maschinenabgeschlossen.

Diese Aussage gilt sogar für abzählbar viele Fairnessbedingungen (ohne Beweis).

Zusammenfassung

- Reaktive Systeme werden formal durch Transitionssysteme modelliert.
- Abläufe eines Transitionssystems Γ sind Folgen $s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \ldots$ von Zuständen und Aktionen,
  so dass s_0 ein Anfangszustand von Γ ist und alle Übergänge $s_i \xrightarrow{A_i} s_{i+1}$ der Zustandsübergangsrelation von Γ entsprechen.
- Zur Systemspezifikation werden Abläufe von Transitionssystemen meist weiter eingeschränkt.
  Insbesondere verwendet man Fairnessbedingungen, um unfaire Abläufe auszuschließen.
  Fairnessbedingungen besagen intuitiv, dass Aktionen, die genügend oft ausführbar sind, immer wieder ausgeführt werden.
- Bei der Analyse von Transitionssystemen ist man hauptsächlich an Aussagen über ihre Abläufe interessiert und identifiziert daher Eigenschaften mit Mengen von Abläufen.
- Eigenschaften können danach klassifiziert werden, wie viel Information durch die Betrachtung endlicher Abläufe gewonnen werden kann. Dabei treffe eine Eigenschaft P auf einen endlichen Ablauf ρ zu, falls es eine unendliche Folge σ mit $\rho \circ \sigma \in P$ gibt.

- Zwei grundlegende Klassen von Eigenschaften sind Sicherheitseigenschaften (“something bad never happens”) und Lebendigkeitseigenschaften (“something good eventually happens”).

  Eine Sicherheitseigenschaft trifft auf σ genau dann zu, wenn sie auf alle endlichen Anfangsstücke
  $\sigma[..n]$ von σ zutrifft. Sicherheitseigenschaften sind also durch endliche Abläufe vollständig bestimmt.

  Eine Lebendigkeitseigenschaft trifft auf alle endlichen Abläufe zu. Endliche Anfangsstücke geben also keinerlei Information auf das Zutreffen der Eigenschaft auf den Gesamtablauf.

- Zu jeder Eigenschaft P gibt es eine kleinste Sicherheitseigenschaft $\mathcal{C}(P) \supseteq P$.
  $\mathcal{C}(P)$ heißt der Sicherheitsabschluss von P.
  Jede Eigenschaft P kann als Durchschnitt einer Sicherheits- und einer Lebendigkeitseigenschaft
  dargestellt werden.

- Ein Paar (S, L) aus einer Sicherheitseigenschaft S und einer beliebigen Eigenschaft L heißt maschinenabgeschlossen, wenn $\mathcal{C}(S \cap L) = S$ gilt.

  In diesem Fall kann jeder endliche Ablauf, der S erfüllt, zu einem Ablauf von $S \cap L$ erweitert werden.
- Transitionssysteme mit schwachen und starken Fairnessbedingungen sind maschinenabgeschlossen.