Modellierung verteilter Systeme

Grundlagen der Programm und Systementwicklung

Sommersemester 2012

Prof. Dr. Dr. h.c. Manfred Broy

Unter Mitarbeit von Dr. M. Spichkova, J. Mund, P. Neubeck

Lehrstuhl Software & Systems Engineering

Teil 5: Ablaufsicht

Prozesse als Abläufe verteilter Systeme

- Ein verteiltes System besteht aus Teilsystemen (genannt Komponenten), die *Aktionen* ausführen
- Aktionen können Teile des Zustands eines Systems ändern (Beispiel: Änderung eines Attributs, Senden und/oder Empfangen einer Nachricht)
- Eine Aktion kann mehrfach im Rahmen des Ablaufs eines System auftreten
  - Jedes Auftreten einer Aktion nennen wir ein *Ereignis*
  - *Man beachte den Unterschied zwischen „Ereignis“ und „Aktion“*
  - *Ein Ereignis ist die Instanz einer Aktion*

Nebenläufige Prozesse und Abläufe

- Gewisse Aktionen/Ereignisse stehen in einer kausalen Beziehung
- Es entstehen
  - nichtsequentielle, „parallele“ Abläufe (Nebenläufigkeit) - wir sprechen von Prozessen oder Aktionsstrukturen
  - stellt man Abläufe (durch „Interleaving“, d.h. durch künstliche Sequentialisierung) als sequenzielle Abläufe dar so spricht man von Spuren (engl. Traces)
- Ein Ablauf besteht aus
  - einer Menge diskreter Ereignisse,
  - die in zeitlicher oder kausaler Beziehung stehen
- Verhalten von System kann durch Mengen von Abläufen (Traces) beschrieben werden

- Anzahl der Abläufe/Aktionen typischerweise groß, oft unendlich

Beschreibung nebenläufiger/sequentieller Abläufe

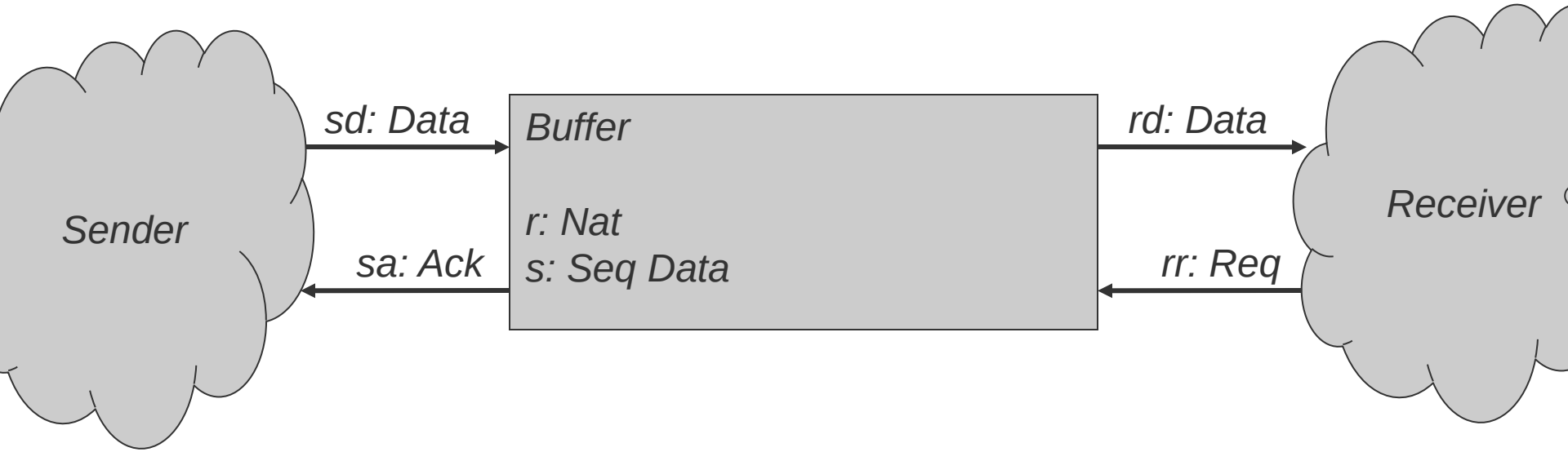
Mit expliziter Nebenläufigkeit

- Beschreibung von Beispielabläufen mit „Message-Sequence-Charts“ (auch Interaktionsdiagramme genannt)
- Prozessdiagramme (In UML: Aktivitätsdiagramme)
- Formale Beschreibung von Abläufen durch:
  - Menge der Aktionen,
  - Menge der Ereignisse und
  - ihre Anordnung in Prozessen („Kausalrelation“)

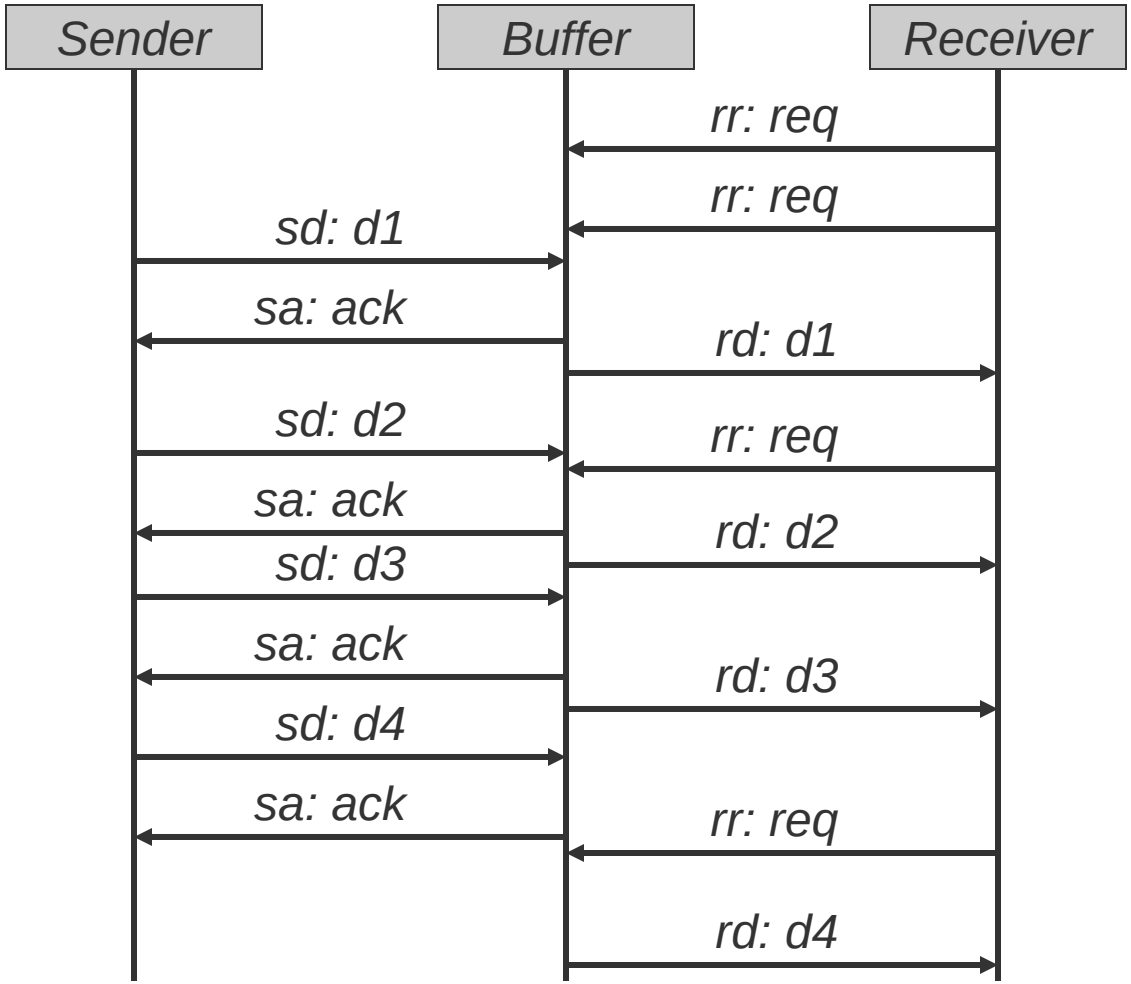
Ohne explizite Nebenläufigkeit (Interleaving, sequentielle Systeme)

- Abläufe repräsentiert durch Ströme von Aktionen

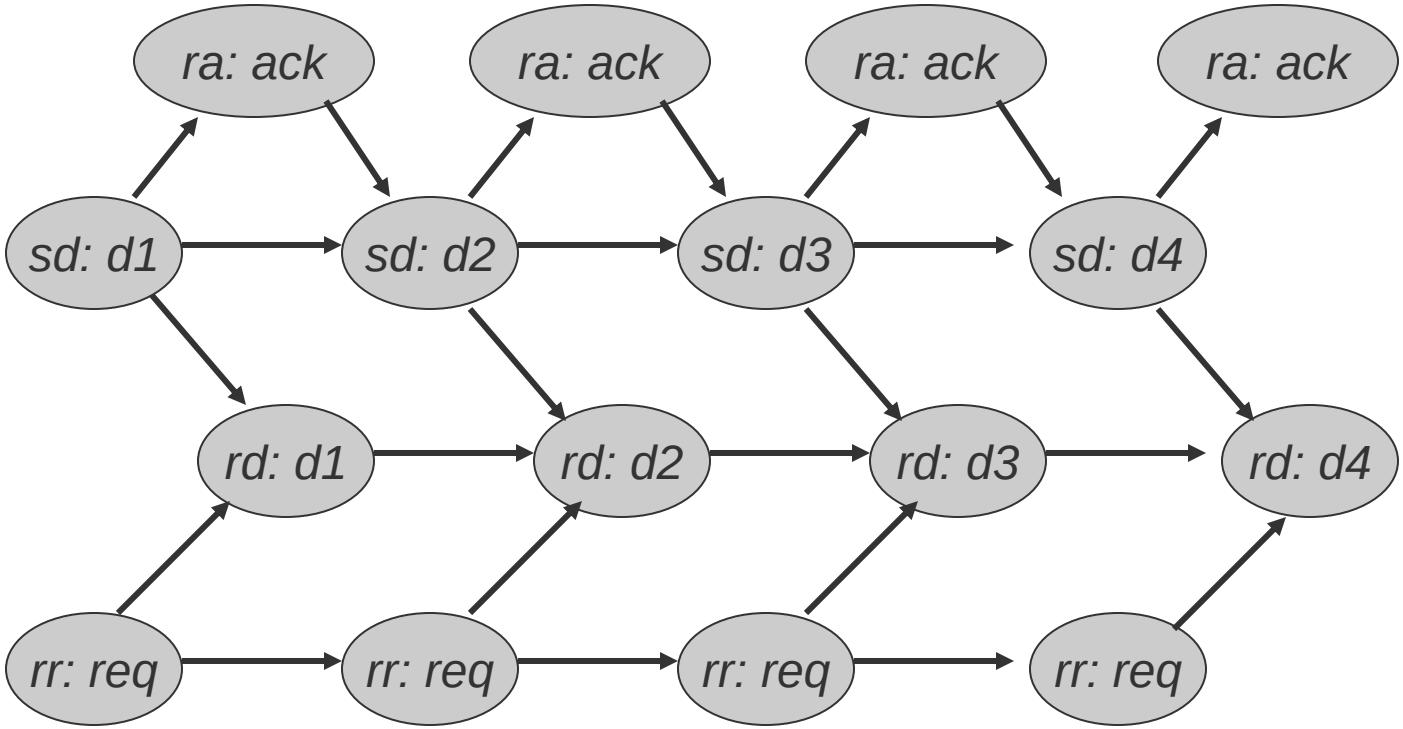
Beispiel: Übertragung über Puffer



Beispiel: Übertragung über Puffer (MSC)



Beispiel: Übertragung über Puffer als Prozess



Aktionsstrukturen (formale Darstellung von Prozessen)

- Menge der Aktionen eines Systems bezeichnen wir mit ***Action***
  - Beschreibung z.B. mit den Mitteln algebraischer Spezifikation durch Angabe einer Sorte
- Aktionen können in einem Ablauf mehrfach auftreten
- Kennzeichnung der einzelnen Instanzen der gleichen Aktion innerhalb eines Ablaufs durch Menge von Ereignissen: ***Event***
  - Wiederum in axiomatischen Spezifikationen beschreibbar durch eine Sorte
- Ablauf eines Systems:

  Partiell (kausal) geordnete Menge ***Event*** von Ereignissen;

  $$\leq : Event \times Event \rightarrow IB$$

  jedem Ereignis ist eine Aktion zugeordnet

  $$act: Event \rightarrow Action$$

Aktionsstrukturen

Definition: *Prozess*

Wir nennen das Paar (≤, act) eine *Aktionsstruktur* oder einen *Prozess* über der Sorte **Event** und bezeichnen mit dom(act) seine Ereignismenge und mit rng(act) seine Aktionsmenge

Man beachte:

Die Kausalbeziehung

$$e_1 \leq e_2$$

bedeutet, dass Ereignis e_2 zwingend e_1 voraussetzt.

Dies impliziert, dass e_1 zeitlich vor e_2 stattfindet.

Die Umkehrung gilt nicht.

Aktionsstrukturen mit Zeit

- Ist für eine Menge **Event** der Ereignisse ein Zeitverhalten angegeben, so sprechen wir von einem gezeiteten Prozess
- Sei ***Time*** die Menge der Zeitpunkte (eine linear geordnete Menge, etwa die Menge der natürlichen Zahlen für *diskrete Zeit* oder die Menge der positiven reellen Zahlen für *kontinuierliche Zeit*)
- Das Zeitverhalten wird angegeben durch zwei Abbildungen

$$\alpha : \textbf{\textit{Event}} \rightarrow \textbf{\textit{Time}}$$

$$\omega : \textbf{\textit{Event}} \rightarrow \textbf{\textit{Time}}$$

mit $\alpha(e) \leq \omega(e)$ für alle Ereignisse e

falls $\alpha(e) = \omega(e)$ gilt sprechen wir von einem punktförmigen Ereignis

Kausale Relation und Zeit

Achtung:

Es gibt einen wichtigen Unterschied zwischen der Zeitrelation und der Kausalrelation

- Wir fordern

$$e \leq e' \Rightarrow \omega(e) \leq \alpha(e')$$

d.h. kausale Abhängigkeit impliziert zeitliches Hintereinander, die Umkehrung stimmt nicht.

- Ein Ereignis e heißt *kausal* für ein Ereignis e‘, wenn Ereignis e‘ nicht stattfinden kann ohne, dass Ereignis e stattgefunden hat.
- Die Kausalitätsrelation eines Prozesses heißt *zeitinduziert*, falls ein Zeitverhalten (α, ω) existiert, so dass

$$e \leq e' \Leftrightarrow (\omega(e) \leq \alpha(e') \vee e = e')$$

Beispiel: Nicht zeitinduzierter Prozess

$$e1 \rightarrow e3$$
$$e2 \rightarrow e4$$

Es gilt: (1) $\omega(e1) \leq \omega(e2)$ *oder*
(2) $\omega(e2) \leq \omega(e1)$
Im Fall (1) gilt $\omega(e1) \leq \alpha(e4)$ da $\omega(e2) \leq \alpha(e4)$
Im Fall (2) gilt $\omega(e2) \leq \alpha(e3)$ da $\omega(e1) \leq \alpha(e3)$

Bedeutung der Kausalrelation

- Was bedeutet es für Ereignisse e, e ‘ (mit e ≠ e ‘, act(e) = a, act(e ‘) = a ‘), wenn
  - die Kausalrelation e ≤ e ‘ gilt, bzw.
  - weder die Kausalrelation e ≤ e ‘ noch e ‘ ≤ e gilt
- Auffassung 1: Kausaler Prozess repräsentiert Menge zeitinduzierter Prozesse

  Gilt weder e ≤ e ‘ noch e ‘ ≤ e, dann besteht keine zwingende zeitliche Festlegung zwischen den Ereignissen; sie können (müssen aber nicht) zeitlich parallel ausgeführt werden.
- Auffassung 2: Kausaler Prozess ist zeitinduzierten Prozess

  Die Kausalitätsrelation entspricht zeitlicher Konstellation

  Achtung: Auffassung 2 erlaubt einen expliziten Begriff von zeitlicher Nebenläufigkeit.

Aktionsstrukturen

- Eine Aktionsstruktur heißt *endlich fundiert*, wenn für jedes Ereignis e die Menge der Ereignisse, die für e kausal sind, endlich ist:

$$\forall e \in dom(act) : \left| \{ e' \in dom(act) : e' < e \} \right| < \infty$$

- Menge aller endlich fundierten Aktionsstrukturen über einer Aktionenmenge *Action* bezeichnen wir mit der Sorte

sort ActStruct Action

- Menge aller Aktionsstrukturen mit endlichen Ereignismengen

$\text{ActStruct}_{\text{fin}}$ Action

- Für endliche Aktionsstrukturen gilt:

$$\left| dom(act) \right| < \infty$$

Aktionsstrukturen

- Vollständiger Ablauf: alle Ereignisse
- Unvollständiger Ablauf: alle Ereignisse bis zu einem bestimmten Zeitpunkt
- *Präfixordnung* auf der Menge der Abläufe:

$$
\begin{aligned}
(\leq, act) \sqsubseteq (\leq', act') \equiv \\
& dom(act) \subseteq dom(act') \wedge \\
& (\leq', act')|_{dom(act)} = (\leq, act) \wedge \\
& \forall e, e' \in dom(act') : e \leq e' \wedge e' \in dom(act) \Rightarrow e \in dom(act)
\end{aligned}
$$

- Gilt $p \sqsubseteq p'$ und $p \neq p'$, dann heißt p *unvollständiger* Ablauf
- Mit endlicher Fundierung hat jeder Prozess kleinste obere Schranke

Eigenschaften von Prozessen

- Jede Aktionsstruktur modelliert Ablauf eines Systems/mögliches Systemverhalten über ein bestimmtes Zeitintervall
- Präfix eines Ablaufs repräsentiert Ablauf über kürzeres Zeitintervall
- Für die Menge der Aktionsstrukturen gilt:
  - Jeder unvollständige Ablauf ist Präfix eines vollständigen Ablaufs
  - Jeder (vollständige) Ablauf ist durch die Menge seiner endlichen Präfixe (unvollständigen Abläufe) eindeutig charakterisiert
- Einem parallelen Programm kann man für einen gegebenen Anfangszustand eine Menge von Abläufen zuordnen, wobei die Aktionen die Einzelanweisungen (Zuweisungen) sind

Eigenschaften von Prozessen

- Menge der Abläufe für gegebene Menge ***Action*** von Aktionen spezifizierbar durch Prädikate:

$$Q : \textbf{\textit{ActStruct Action}} \rightarrow \textbf{\textit{IB}}$$

- Zwei Klassen von Eigenschaften bei Prädikaten über Prozesse
  - *Sicherheitsbedingungen*: schließen bestimmte unerwünschte, bereits an endlichen Teilprozessen beobachtbare Verhaltensmuster aus
  - *Lebendigkeitsbedingungen*: stellen sicher, dass bestimmte Aktionen schließlich auftreten
- Systemeigenschaften meist Mischung aus Sicherheits- und Lebendigkeitseigenschaften
- Jede Systemeigenschaft lässt sich in eine reine Sicherheits- und eine reine Lebendigkeitseigenschaft aufspalten

Eigenschaften von Prozessen

- Sicherheitsbedingungen (*Safety Properties*)
  - Formuliert durch Prädikate auf der Menge der Abläufe, die für alle endlichen (unvollständigen) Teilabläufe gelten
- Lebendigkeitsbedingungen (*Liveness Properties*)
  - Formuliert durch Prädikate, die für die vollständigen Abläufe gelten müssen
- (permanente) *Invariante*
  - Prädikat gilt für alle endlichen (Präfix-)Abläufe eines Systems

- Sicherheitsbedingungen lassen sich als permanente Invariante deuten

Eigenschaften von Prozessen

Ein Prädikat Q : $\boldsymbol{ActStruct\ Action \rightarrow IB}$

heißt (reine) *Sicherheitseigenschaft*, wenn für alle Prozesse p gilt:

$$Q(p) = (\forall\ p\text{‘} \in \boldsymbol{ActStruct_{fin}\ Action}: p\text{‘} \sqsubseteq p \Rightarrow Q(p\text{‘}))$$

Ein Prädikat Q heißt (reine) *Lebendigkeitseigenschaft*, wenn gilt

$$\forall\ p\text{‘} \in \boldsymbol{ActStruct_{fin}\ Action}: \exists\ p \in \boldsymbol{ActStruct\ Action}: p\text{‘} \sqsubseteq p \wedge Q(p)$$

Traces: Sequentielle Prozesse als Ströme von Aktionen

- Ein sequentieller Prozess besteht aus einer linear geordneten Menge von Ereignissen
- Jeder sequentielle Prozess kann als Strom von Aktionen dargestellt werden
- Ist für eine Aktionsstruktur (≤, act) die Kausalitätsordnung ≤ linear, so lässt sie sich auch durch trace(≤, act) ∈ Stream Action darstellen:

$$\mathrm{dom}(act) = \varnothing \Rightarrow \mathrm{trace}(\leq, act) = \langle\rangle,$$
$$e \in \mathrm{dom}(act) \wedge (\forall\, e' \in \mathrm{dom}(act) : e \leq e') \Rightarrow$$
$$\mathrm{trace}(\leq, act) = act(e)\ \&\ \mathrm{trace}(\leq, act)|_{\mathrm{dom}(act)\setminus\{e\}})$$

- Ein Strom von Aktionen kann somit als Spezialfall einer Aktionsstruktur aufgefasst werden

Von Prozessen zu Traces durch Interleaving

- Ein Prozess $p' = (\leq', act')$ heißt Interleaving für einen Prozess $p = (\leq, act)$ falls
  - $dom(p') = dom(p)$ und $act' = act$
  - die Kausalordnung auf p' linear ist
  - für alle $e, e' \in dom(p)$: $e \leq e' \Rightarrow e \leq' e'$

- Gegeben eine Menge P von Prozessen, dann definieren wir die Menge der Traces traces(P) wie folgt

  $traces(P) = \{trace(\leq', act'): \exists\, p \in P: (\leq', act') \text{ Interleaving von } p\}$

Abläufe als Zustandsfolgen und temporale Logik

- Ströme von Zuständen und/oder Aktionen entsprechen sequentiellen Abläufen
- Eigenschaften von Strömen lassen sich durch Prädikate beschreiben

$$Q : \textbf{\textit{Stream Action}} \rightarrow \textbf{\textit{IB}}$$

- Besondere Technik der Formulierung von Prädikaten über unendlichen Zustands-/Aktionsströmen: *temporale Logik*
- Wir betrachten *„Linear Time Temporal Logic“*, die über Sequenzen von Systemzuständen spricht

Abläufe als Zustandsfolgen und temporale Logik

Sei α eine Sorte und sei Q ein Prädikat über der Menge Stream α

$Q: Stream\ \alpha \rightarrow Bool$

Wir erhalten neue Prädikate über der Menge *Stream α* durch die folgenden Festlegungen

$$
\begin{array}{lcll}
(\circ\ \mathsf{Q})(\mathsf{s}) & \equiv & \mathsf{Q}(\mathsf{rest}(\mathsf{s})) & Nexttime \\
(\Diamond\ \mathsf{Q})(\mathsf{s}) & \equiv & \exists\, i \in \mathbb{N} : \mathsf{Q}(\mathsf{rest}^i(\mathsf{s})) & Sometimes \\
(\Box\ \mathsf{Q})(\mathsf{s}) & \equiv & \forall\, i \in \mathbb{N} : \mathsf{Q}(\mathsf{rest}^i(\mathsf{s})) & Always \\
(\mathsf{Q}\ \textbf{until}\ \mathsf{Q}')(\mathsf{s}) & \equiv & \exists\, i \in \mathbb{N} : \forall\, j \in \mathbb{N} : (j < i \Rightarrow \mathsf{Q}(\mathsf{rest}^j(\mathsf{s}))) \wedge \mathsf{Q}'(\mathsf{rest}^i(\mathsf{s})) &
\end{array}
$$

Abläufe als Zustandsfolgen und temporale Logik

- Für jedes Prädikat Q auf Strömen ist ◦ Q, ◊ Q und □ Q wieder ein Prädikat auf Strömen
- Sind Q und Q´ Prädikate auf Strömen, so ist auch Q **until** Q´ ein Prädikat auf Strömen
- Die Symbole ◦, ◊, □ und **until** können wir als Prädikattransformatoren auf Prädikaten für Ströme auffassen

$$R : \alpha \rightarrow Bool$$

- Wir erweitern Prädikate R über einer Sorte α zu Prädikaten über der Sorte Stream α durch folgende Festlegung

$$R(s) = R(\text{first}(s))$$

Somit erhalten wir für gegebene Prädikate auf der Menge α Prädikate und temporallogische Ausdrücke für Ströme

Abläufe als Zustandsfolgen und temporale Logik

Zusicherungen über Σ definieren Menge von Zuständen

Ensprechen Prädikaten:

$$Q : \Sigma \rightarrow Bool$$

Aussagen über Ströme von Zuständen mit temporallogischen Formeln

$x \geq y$	Aussage $x \geq y$ gilt für ersten Zustand im Strom
$\circ\, x \geq y$	Aussage $x \geq y$ gilt für zweiten Zustand im Strom
$\Box\, x \geq y$	Aussage $x \geq y$ gilt für alle Zustände im Strom
$\Diamond\, x \geq y$	Aussage $x \geq y$ gilt irgendwann im Strom
$x \geq y$ **until** $x = 0$	Irgendwann gilt $x = 0$ und davor gilt bis dahin $x \geq y$

Abläufe als Zustandsfolgen und temporale Logik

- *leads-to*-Eigenschaft: Q **leads_to** P

  temporallogisch:

$$\Box(Q \Rightarrow \Diamond P)$$

Falls irgendwann im Strom Q gilt, so gilt irgendwann danach auch P

- Stabilität:

  Falls P gilt, so gilt P von da an stets

$$\Box(P \Rightarrow \Box P)$$

Abläufe als Zustandsfolgen und temporale Logik

Formulierung gewisser Eigenschaften eines Stroms

(1) Prädikat Q führt auf R (*Lebendigkeitsbedingung*)

$$\Box\ (Q \Rightarrow \Diamond\ R)$$

(2) Prädikat R erfordert Q (*Sicherheitsbedingung*)
entspricht folgender Aussage über den Strom s:

$$\forall\ s', a : |s'| < \infty\ \wedge\ s' \frown \langle a \rangle \sqsubseteq s\ \wedge\ R(a)\ \Rightarrow\ \exists\ s'', b : s'' \frown \langle b \rangle \sqsubseteq s'\ \wedge\ Q(b)$$

Koordination, gegenseitiger Ausschluss

- Gewisse Aktionen verteilter Systeme können kausal *unabhängig* ausgeführt werden
- Zwei Fälle bei nicht unabhängigen Aktionen
  - Gegenseitiger Ausschluss

    $act(e) = a \wedge act(e') = b \Rightarrow (e < e' \vee e' < e)$

  - Echte Kausalität auf Aktionen

    Aktion a heißt eins-zu-eins kausal für Aktion b, wenn für alle Präfixe q = (≤, act) unvollständiger Abläufe gilt #(q,a) ≥ #(q,b), wobei #(q,a) der Anzahl der Ereignisse in q, die mit der Aktion a markiert sind, entspricht

    $$|\{e \in dom(act) : act(e) = a\}| \geq |\{e \in dom(act) : act(e) = b\}| \qquad Sicherheit$$

    für alle vollständigen Abläufe:

    $$|\{e \in dom(act) : act(e) = a\}| = |\{e \in dom(act) : act(e) = b\}| \qquad Lebendigkeit$$

Koordination, gegenseitiger Ausschluss

- Für Zustandsfolgen mit Zuständen, in denen wir Zähler für die Zahl der Aktionen haben

  Für alle k ∈ IN:

  $\Box\ (\#\mathsf{a} > k \Rightarrow \Diamond\ \#\mathsf{b} > k)$ $\quad Lebendigkeit$

  $\Box\ \#\mathsf{a} \geq \#\mathsf{b}$ $\quad Sicherheit$

Koordination, gegenseitiger Ausschluss

- Weitere Beschreibungstechniken zur Wiedergabe von Abläufen verteilter Systeme:
  - UML (Unified Modeling Language): Aktions- und Interaktionsdiagramme
  - SDL (Specification and Description Language): Sequenzdiagramme (Message Sequence Charts)
  - ARIS: Prozessdiagramme

Zusammenhang zwischen stromverarbeitenden Funktionen und Abläufen

- Stromverarbeitende Funktion
  - Abbildung von Eingabedatenströme auf Ausgabedatenströme
  - Jedes Paar (x, y) von Eingabeströmen x und Ausgabeströmen y beschreibt mögliche Eingabe und dadurch stimulierte Systemausgabe y
- Ein- und Ausgabe werden stückweise in der Interaktion zwischen System und Umgebung bestimmt
- Diese Interaktion kann explizit gemacht werden
  - Zusammenmischen von Eingaben x und Ausgaben y in einen Strom
  - Wir erhalten einen Ablauf von Ein- und Ausgaben
  - Kausalität zwischen Ein- und Ausgabe durch Monotonie der Funktion

Zusammenhang zwischen stromverarbeitenden Funktionen und Abläufen

- Für jede stromverarbeitende Funktion lässt sich mit jeder Eingabe eine Menge M von Strömen verbinden, die Mischung aus Ein- und Ausgabe bilden
- sei

$$f: M^{\omega} \rightarrow M^{\omega}$$

eine Abbildung auf Strömen und

$$pfilter, nfilter: IB^{\omega} \times M^{\omega} \rightarrow M^{\omega}$$

- Abbildungen mit

$$pfilter(\ true\ \&\ z, m\ \&\ x) = m\ \&\ pfilter(z, x)$$
$$pfilter(\ false\ \&\ z, m\ \&\ x) = pfilter(z, x)$$
$$nfilter(\ true\ \&\ z, m\ \&\ x) = nfilter(z, x)$$
$$nfilter(\ false\ \&\ z, m\ \&\ x) = m\ \&\ nfilter(z, x)$$

Zusammenhang zwischen stromverarbeitenden Funktionen und Abläufen

- Für jede stromverarbeitende Funktion lässt sich mit jeder Eingabe eine Menge M von Strömen verbinden, die Mischung aus Ein- und Ausgabe bilden
- Strom s ist Ablauf von f zu Eingabe $x \in M^{\omega}$, wenn gilt:

  Es existiert ein Boolescher Strom $z \in IB^{\omega}$ mit

  (1) Der Ablauf ist Mischung aus Ein- und Ausgabe

  $$\mathrm{pfilter}(z, s) = x \wedge \mathrm{nfilter}(z, s) = f(x)$$

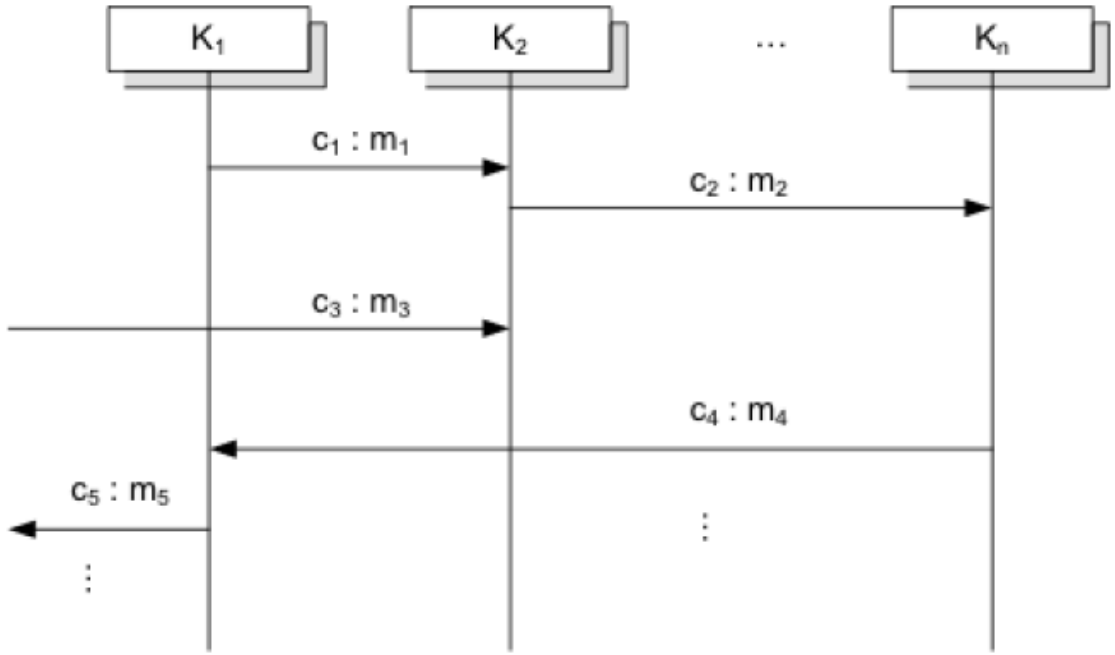
  (2) Die Ausgabe kommt nach der erforderlichen Eingabe

  $$\forall\, s' : s' \sqsubseteq s \Rightarrow \mathrm{nfilter}(z, s') \sqsubseteq f(\mathrm{pfilter}(z, s'))$$

  (1) führt auf Ströme, in denen Ein- und Ausgabeströme gemischt sind, die Eingabewerte jedoch so früh, wie nach (1) und (2) nur möglich, auftreten

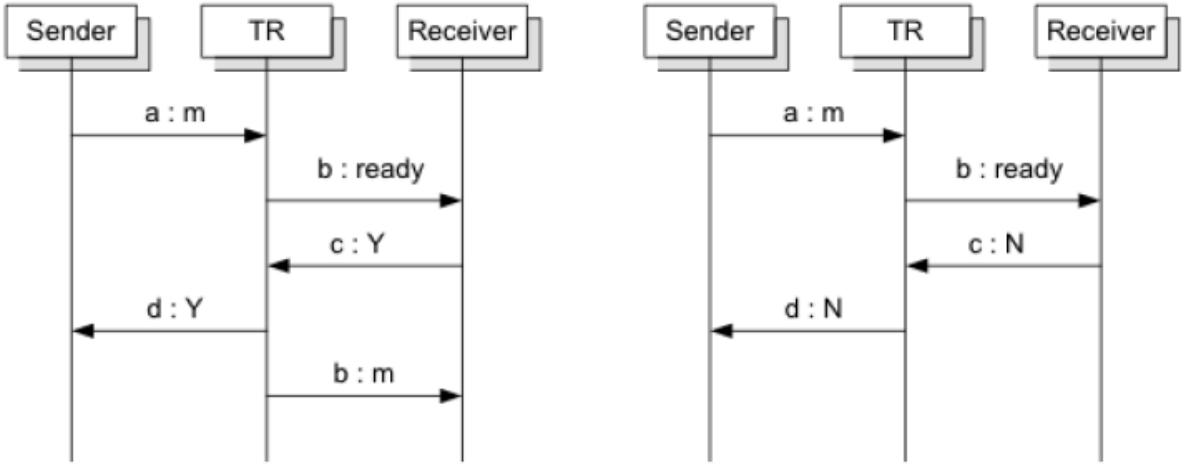
Abläufe verteilter kommunizierender Systeme

- Interaktionsdiagramme (*Message Sequence Charts, Event Traces*)
  - Beschreiben einen Ablauf eines Systems durch Angabe der Nachrichten, die zwischen den Komponenten ausgetauscht werden



Abläufe verteilter kommunizierender Systeme

- Schematische Interaktionsdiagramme können in Gleichungen abgebildet werden



$$(1) \quad f_{TR}(\langle a : m \rangle) = \langle b : ready \rangle$$

$$(2) \quad f_{TR}(\langle a : m \rangle \frown \langle c : Y \rangle \frown x) = \langle b : ready \rangle \frown \langle d : Y \rangle \frown \langle b : m \rangle \frown f_{TR}(x)$$

$$(3) \quad f_{TR}(\langle a : m \rangle \frown \langle c : N \rangle \frown x) = \langle b : ready \rangle \frown \langle d : N \rangle \frown f_{TR}(x)$$

Abläufe verteilter kommunizierender Systeme

- Interaktionsdiagramme
  - Form der logischen Spezifikation durch Gleichungen für stromverarbeitende Funktionen
  - Erlauben anschauliche Darstellung der Interaktion zwischen den Komponenten eines Systems
  - Bei komplizierten Systemen besteht die Gefahr, dass eine große Anzahl von Interaktionsdiagrammen notwendig ist, um das Systemverhalten umfassend zu beschreiben