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

Schnittstellensicht: Diskrete Datenströme und Datenflussfunktionen

Schnittstelle

- Eine **Schnittstelle** ist festgelegt durch
  - eine Grenze zwischen einem System und seiner Umgebung
  - eine Grenze zwischen Systemteilen
- Durch die Schnittstelle wird ein System von seiner **Umgebung** abgegrenzt
- Die **Schnittstelle** wird gegeben durch
  - die Angabe, welche Arten von Informationen (allgemeiner: Material, Energie etc.) zwischen System und Umgebung ausgetauscht werden (für diskrete Systeme die Aktionen an der Schnittstelle
  - die **Wechselwirkungen** zwischen System und Umgebung (gegeben etwa durch die Menge der Folgen von Aktionen)

Schnittstellensicht

- **Schnittstelle**
  - Alle relevanten Informationen über das Zusammenwirken des Systems mit seiner Umgebung
- **die Schnittstellensicht (Außenansicht) konzentriert sich auf die Beschreibung der Schnittstelle(n) eines Systems**
  - Interaktionsmuster, die bei Nutzung des Systems auftreten
  - Welche Informationen zwischen dem System und seiner Umgebung ausgetauscht werden
  - Wie System und Umgebung zusammenwirken
- **Schnittstellenabstraktion**
  - Übergang von der internen Sicht zur Schnittstellensicht eines Systems

Schnittstellenkompatibilität

- Ein System A heißt zu einem System B **schnittstellenkompatibel** wenn A in beliebigen Umgebungen durch B ersetzt werden kann, ohne, dass sich das Verhalten aus Sicht der Umgebung ändert.

- **Schnittstellenkompatibilität (System A stets durch System B ersetzbar ohne nach sichtbare Verhaltensänderung):**
  - keine Äquivalenzrelation
  - eine partielle Ordnung

- **Schnittstellenäquivalenz**
  - A schnittstellenkompatibel zu B ist und umgekehrt
- **Syntaktische Schnittstelle (auch statische Schnittstelle):**
  - Festlegung, in welcher grundsätzlichen Form ein Informationsaustausch möglich ist

Programmiertechnisch: Schnittstelle

- **Syntaktische (statische) Schnittstelle** einer Komponente:

  Bestimmt, in welche Umgebungen ein Programm eingefügt werden kann, ohne dass es syntaktische Probleme gibt
- **Semantische (dynamische) Schnittstelle (Verhalten)** einer Komponente:

  Bestimmt die Interaktion (den Informationsaustausch) zwischen der Komponente und seiner Umgebung

Formen von Schnittstellen

- Gemeinsamer Speicher (vgl. parallele Komposition von Zustandsmaschinen mit überlappenden Zuständen)
- Informationsaustausch durch Nachrichten/Signale

Syntaktische Schnittstellen zum Nachrichtenaustausch: Kanäle

Ein **Kanal** ist ein Medium für die Übertragung von Nachrichten:

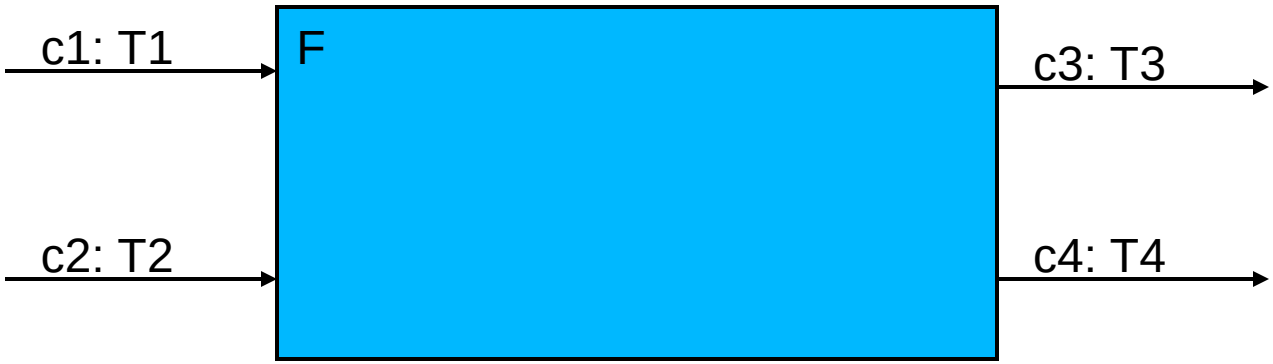
- Der Datentyp (die Sorte) des Kanals gibt an, welche Nachrichtentypen übertragen werden
- Der Kanal überträgt Nachrichten
  - sequentiell (eine Nachricht nach der anderen)
  - gerichtet (nur in einer Richtung) vom Sender zum Empfänger

Bemerkung:

- Es gibt eine Reihe unterschiedlicher Konzepte (Protokolle), wie die Übertragung der Nachrichten erfolgt (verzögert, gepuffert, synchron, asynchron, ...)
- Wir betrachten zunächst eine elementare Form der Nachrichtenübertragung

Systeme mit Ein/Ausgabekanälen

- Ein System F mit Ein/Ausgabekanälen besitzt
  - eine Menge von Eingabekanälen I
  - eine Menge von Ausgabekanälen O
  - jeder Kanal in I und O hat einen Identifikator und einen Typ
- Mit (I ▸ O) bezeichnen wir die **syntaktische Schnittstelle** des Systems



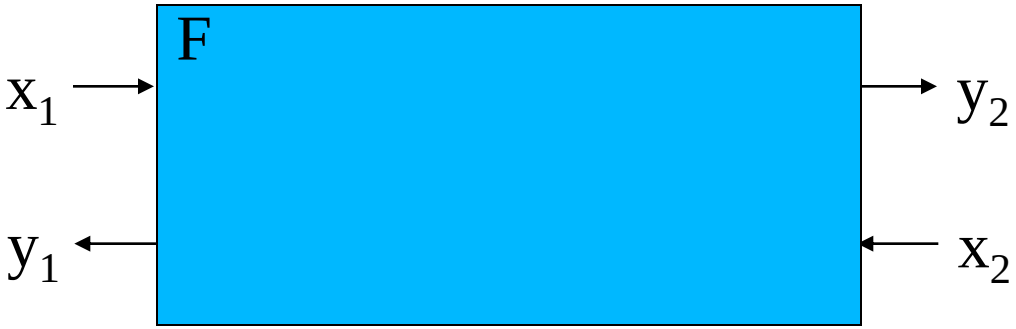
Zustandsmaschinen mit Ein- und Ausgabe über Kanäle

- Eine nichtdet. ZM (Δ, Σ_0) mit Ein- und Ausgabe über die Kanäle der syntaktische Schnittstelle (I ▸ O) benutzt als Eingaben Belegungen der Eingabekanäle und als Ausgaben Belegungen der Ausgabekanäle
- Eine Belegung x der Menge C von Kanälen ordnet jedem Kanal in C die übertragenen Nachrichten zu
- Sei M die Menge der Nachrichten; wir betrachten folgende Arten von Belegungen der Kanäle in C
  - $x: C \rightarrow M$ Genau eine Nachricht pro Kanal
  - $x: C \rightarrow M \cup \{-\}$ Höchstens eine Nachricht pro Kanal
  - $x: C \rightarrow M^*$ Eine Sequenz von Nachrichten pro Kanal
- Mit B[C] bezeichnen wir die Menge der Belegungen der Kanäle in C mit Sequenzen
- Die ZM mit Schnittstelle (I ▸ O) besteht dann
  - aus einer Zustandsmenge Σ,
  - Anfangszuständen $\Sigma_0 \subseteq \Sigma$ (oder einem $\sigma_0 \in \Sigma$) und
  - einer Zustandsübergangsfunktion $\Delta: \Sigma \times B[I] \rightarrow \wp(\Sigma \times B[O])$

Tabelle für Berechnung einer Zustandsmaschine

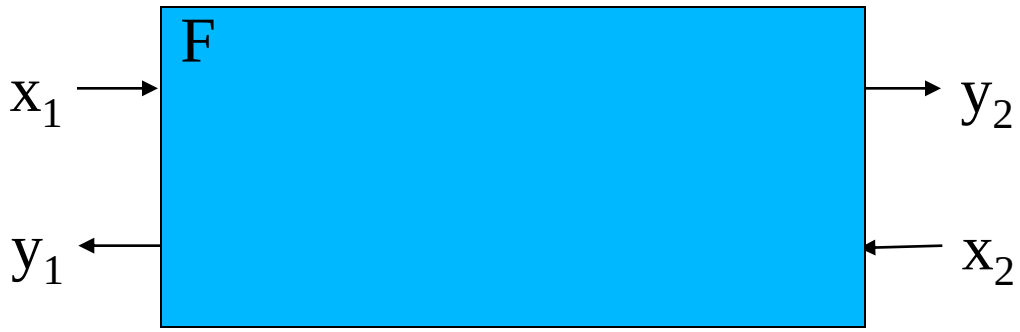
	y_2	1	1	2	1	2	1	1	...
	y_1	-4	3	-5 -9	-3	8 10	7	11	...
	x_2	4		8 4			3		...
	x_1		7		6	11 2		4	...
σ_1	0	1	2	4	5	7	8	9	...
σ_2	0	-4	3	-9	-3	10	7	11	...
	0	1	2	3	4	5	6	7	8

time



Schnittstellenabstraktion

	y_2	1	1	2	1	2	1	1	...
	y_1	-4	3	-5 -9	-3	8 10	7	11	...
	x_2	4		8 4			3		...
	x_1		7		6	11 2		4	...
	0	1	2	3	4	5	6	7	8 time



Schnittstellenverhalten

- Für ein System mit Ein- und Ausgabe über die Kanäle der syntaktische Schnittstelle (I ▸ O) definieren wir das **Schnittstellenverhalten** durch die Abbildung der Kanalgeschichten der Eingabekanäle auf die Kanalgeschichten der Ausgabekanäle
- Eine Kanalgeschichte entspricht der **Belegung** der Kanäle und der Ausgaben durch Datenströme
- Ein Datenstrom ist eine Folge (Sequenz) von Nachrichten

Unbeschränkte Datenströme

- Mit IN = {0, 1, 2, ... } bezeichnen wir die natürlichen Zahlen
- Mit IN_+ = {1, 2, ... } bezeichnen wir die echt positiven natürlichen Zahlen

- Sei M die Menge der Nachrichten;

  wir betrachten folgende Arten von unbeschränkten Strömen

  - s: $IN_+ \rightarrow M$ Unendlicher Strom von Nachrichten aus M
  - s: $IN_+ \rightarrow (M \cup \{-\})$ Unendlicher Strom von Nachrichten aus M mit leeren Stellen
- Mit M^∞ bezeichnen wir die Menge der unendlichen Ströme
- Mit $(M^*)^\infty$ bezeichnen wir die Menge der unendlichen Ströme von Sequenzen von Nachrichten aus M
  - s: $IN_+ \rightarrow M^*$ Eine Sequenz von Nachrichten pro Kanal

Diskrete Zeit

- Ein Strom

$$s: IN_+ \rightarrow M^*$$

  kann als Kommunikationsgeschichte eines Kanals über der diskreten Zeit IN_+ gesehen werden.
- Die Zeit wird durch eine Folge von **Zeitintervallen** dargestellt.
- Jede Zahl $t \in IN_+$ bezeichnet dann ein Zeitintervall.
- Für jede Zeit $t \in IN_+$ bezeichnet dann s(t) die Sequenz der Nachrichten, die im Zeitintervall t gesendet wurde.

s	⟨a b a b⟩	⟨ ⟩	⟨c c a⟩	⟨a a b c⟩	⟨b⟩	⟨d e⟩	⟨e d⟩	...
0	1	2	3	4	5	6	7	8

time

Belegungen von Kanälen mit Strömen

- Eine **Belegung** x einer Menge C von Kanälen ordnet jedem Kanal in C den Strom der übertragenen Nachrichten zu
- Sei M die Menge der Nachrichten; wir betrachten folgende Arten von Belegungen
  - $x: C \rightarrow (IN_+ \rightarrow M)$ Genau eine Nachricht
  - $x: C \rightarrow (IN_+ \rightarrow M \cup \{-\})$ Höchstens eine Nachricht
  - $x: C \rightarrow (IN_+ \rightarrow M^*)$ Eine Sequenz von Nachrichten
- Mit IH[C] oder mit $\dot{C}$

  bezeichnen wir die Menge der Belegungen der Kanäle mit Strömen, die den Typen der Kanäle entsprechen

Schnittstellenverhalten: Verhaltensfunktionen

- Für ein System mit Ein- und Ausgaben über die Kanäle der syntaktischen Schnittstelle

$$(I \triangleright O)$$

definieren wir sein **Schnittstellenverhalten** f durch die Abbildung der Kanalgeschichten der Eingabekanäle auf die Kanalgeschichten der Ausgabekanäle

$$f : \dot{I} \rightarrow \dot{O}$$

Durch f wird jeder Eingabehistorie x (Belegung der Eingabekanäle durch Ströme) eine Ausgabehistorie y = f(x) (Belegung der Ausgabekanäle durch Ströme)

- Die Eigenschaften der Verhaltensfunktionen betrachten wir später.

Verhaltensmodell: Kanäle und Ströme

C	Menge von typisierte Kanälen
type: C → TYPE	Typzuordnung
x: C → (IN_+ → M*)	Kanalgeschichte für Nachrichten vom Typ M
$\vec{C}$ oder IH[C]	Menge der Kanalgeschichten für Kanäle aus C

Schnittstellenmodell für Systeme

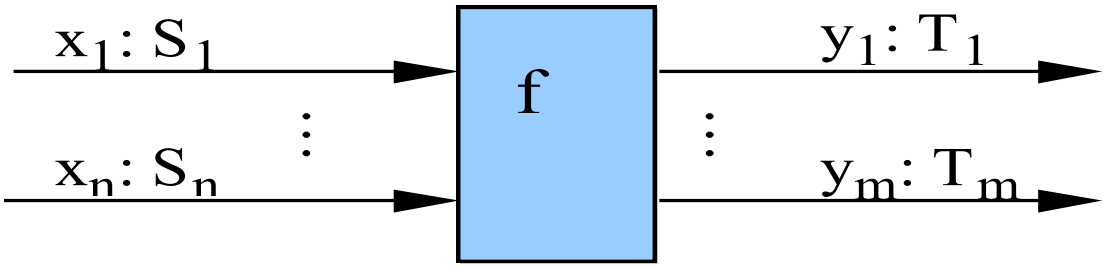
$I = \{ x_1 , x_2 , \ldots \}$ Menge der Eingabekanäle

$O = \{ y_1 , y_2 , \ldots \}$ Menge der Ausgabekanäle

Syntaktische Schnittstelle: $(I \blacktriangleright O)$

Schnittstellenverhalten

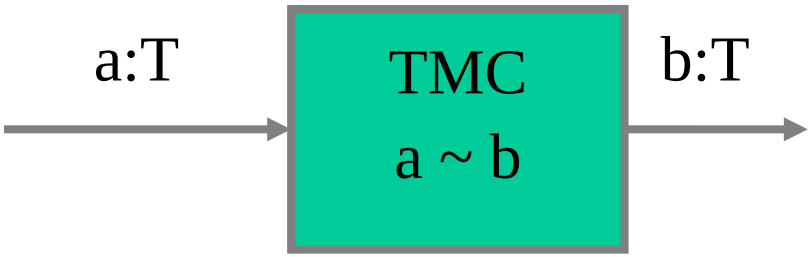
$$f : \vec{I} \rightarrow \vec{O}$$



Beispiel: Spezifikation eines Systems

Übertragungskomponente TMC

TMC
in a: T **out** b: T
a ~ b



$$a \sim b \equiv (\forall\ m \in T: \{m\}\#a = \{m\}\#b)$$

{m}#a bezeichnet die Anzahl der Vorkommnisse von m im Strom a

Die Zusicherung a ~ b heißt Schnittstellenzusicherung

Schnittstellenzusicherung

- Eine Funktion f mit $f : \dot{I} \rightarrow \dot{O}$

  erfüllt die Spezifikation mit Schnittstellenzusicherung Q, wenn gilt

$$\forall\ x, y: y = f(x) \Rightarrow Q[x(i_1)/i_1\ , \ldots, x(i_m)/i_m\ , y(o_1)/o_1\ , \ldots, y(o_n)/o_n]$$

  wobei $I = \{i_1, \ldots, i_m\}$, $O = \{o_1, \ldots, o_n\}$

Verifikation von Eigenschaften

Aus Schnittstellenspezifikationen lassen sich Eigenschaften beweisen

- Sicherheit

$$\{m\}\#b > 0 \wedge TMC(a, b) \Rightarrow \{m\}\#a > 0$$

- Lebendigkeit

$$\{m\}\#a > 0 \wedge TMC(a, b) \Rightarrow \{m\}\#b > 0$$

Schnittstellenabstraktion für Zustandsmaschinen

Eine Zustandsmaschine $(\otimes, \subset)$ mit synt. Schnittstelle $(I \blacktriangleright O)$ besteht aus

- $\mathbb{C}$ Zustandsmenge
- $\subset \square \mathbb{C}$ Anfangszustände
- Transitionsfunktion:

$$\otimes : \mathbb{C} \times \oplus \square \mathbb{C} \times \mathfrak{I}$$

in nichtdeterministischer Version:

$$\otimes : \mathbb{C} \times \oplus \square \wp(\mathbb{C} \times \mathfrak{I})$$

wobei:

$$E = I \square M^*$$

$$A = O \square M^*$$

Berechnungen

Eine Zustandsmaschine (⊗, ⊄) definiert für jeden Anfangszustand

$$\int_0 \square \not\subset$$

und jede Folge von Eingaben

$$e_1, e_2, e_3, \ldots \square E$$

eine Folge von Zuständen

$$\int_1, \int_2, \int_3, \ldots \square \copyright$$

und eine Folge von Ausgaben

$$a_1, a_2, a_3, \ldots \square A$$

durch

$$(\int_{i+1}, a_{i+1}) \square \otimes(\int_i, e_{i+1})$$

Ist die Zustandsmaschine (⊗, ⊄) deterministisch und total, dann ist die Folge der Zustände und die Folge der Ausgaben durch den Anfangszustand und die Folge der Eingaben eindeutig bestimmt.

Schnittstellenabstraktion für det. Zustandsmaschinen

- Wir erhalten Berechnungen

$$\sigma_0 \xrightarrow{e_1/a_1} \sigma_1 \xrightarrow{e_2/a_2} \sigma_2 \xrightarrow{e_3/a_3} \sigma_3 \ldots$$

- Für einen Anfangszustand σ' definieren wir die Funktion

$$f : \dot{I} \rightarrow \dot{O}$$

durch

$$f_{\sigma'}(x) = y \Leftrightarrow \exists\, \sigma_i : \sigma' = \sigma_0 \wedge \forall\, i \in IN : (\sigma_{i+1}, x(i+1)) = \Delta(\sigma_i, y(i+1))\}$$

- $f_{\sigma'}$ heißt die Schnittstellabstraktion der Zustandsübergangsfunktion .

$$Abs((\Delta, \sigma')) = f_{\sigma'}$$

Ströme (1)

- Entsprechen endlichen und unendlichen Sequenzen von Datenelementen (Signalen, Nachrichten, Aktionen, Ereignissen, Zuständen)
- Dienen zur Darstellung der Folge von Datenelementen, die über ein sequentielles Kommunikationsmedium zur Übertragung geschickt werden
- Sorte α mit Trägermenge $M^{\perp} = M \cup \{ \perp \}$

  $\perp$ steht für „undefiniert“

  Es gelte $M = M^{\perp} \setminus \{ \perp \}$
- Stream α hat die Trägermenge M^{ω}

$$M^{\omega} = M^{*} \cup M^{\infty}$$

Stromverarbeitende Funktionen (1): Operationen

linksstrikt

$\_\&\_ : \alpha \times \text{Stream } \alpha \rightarrow \text{Stream } \alpha$

$\text{rest} : \text{Stream } \alpha \rightarrow \text{Stream } \alpha$

$\text{first} : \text{Stream } \alpha \rightarrow \alpha$

$\langle\rangle : \text{Stream } \alpha$

$\_\hat{}\_ : \text{Stream } \alpha \times \text{Stream } \alpha \rightarrow \text{Stream } \alpha$

Axiome für $x \in M$

$\text{first}(x \,\&\, s) = x$

$\text{rest}(x \,\&\, s) = s$

$\perp \,\&\, s = \langle\rangle$

$(x \,\&\, s_1) \hat{}\, s_2 = x \,\&\, (s_1 \hat{}\, s_2)$

Zeitabstraktion

- Gegeben ein Strom

$$x: IN_+ \rightarrow M^*$$

der aus einer Folge von Sequenzen besteht.

Wir definieren die Zeitabstraktion

$$timeabs: (IN_+ \rightarrow M^*) \rightarrow M^\omega$$

wie folgt:

$$timeabs(s) = s(1)\hat{}s(2)\hat{}s(3) \ldots$$

- Die Zeitabstraktion timeabs(s) ist
  - endlich, genau dann, wenn in s nur für endlich viele $t \in IN_+$ nicht leer ist

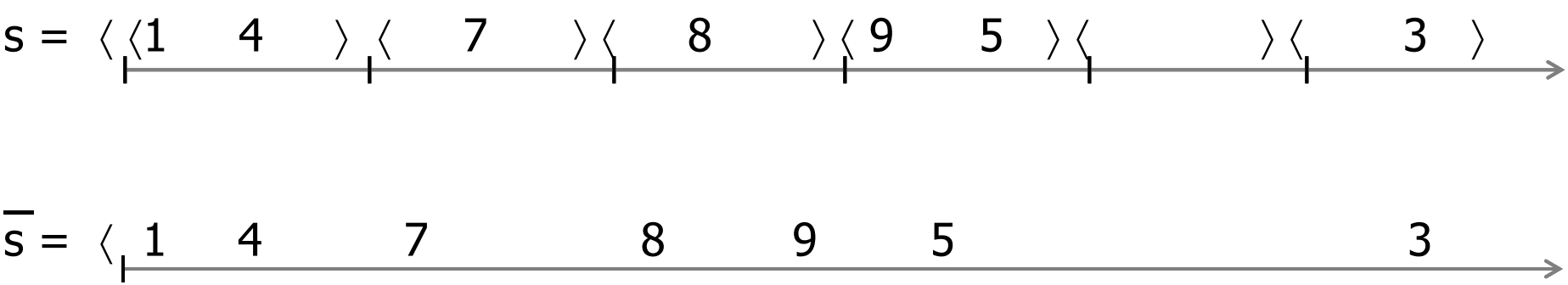
Zeitabstraktion: Notation

- Abkürzend schreiben wir:

$$timeabs(s) = \bar{s}$$

$\bar{s}$ bezeichnet die Zeitabstraktion von s

Time Abstraction



Zeitunabhängigkeit

- Eine Verhaltensfunktion

$$f : I^{'} \rightarrow O^{'}$$

heißt **zeitunabhängig**, wenn für alle x, x‘ und y, y‘ gilt

$$f(x) = y \wedge f(x') = y' \wedge \forall\, c \in I: timeabs(x(c)) = timeabs(x'(c)) \Rightarrow$$
$$\forall\, c \in O: timeabs(y(c)) = timeabs(y'(c))$$

d.h. die Nachrichtenfolge timeabs(y(c)) in den Ausgabekanälen $c \in O$ hängt nur von den Nachrichtenfolgen in den Eingabekanälen, aber nicht von deren Eingabezeitpunkten ab.

Zeitunabhängigkeit

- Ist f zeitunabhängig,

$$f : \dot{I} \rightarrow \dot{O}$$

dann existiert zu $I = \{i_1, ..., i_n\}$ für jeden Ausgabekanal $c \in O$ eine Funktion auf Strömen

$$f' : M^{\omega} \times ... \times M^{\omega} \rightarrow M^{\omega}$$

- mit
  - $f(x) = y \Rightarrow$
  - $f'(timeabs(x(i_1)), ... , timeabs(x(i_n))) = timeabs(y(c))$
- f‘ heißt dann Zeitabstraktion für f

Stromverarbeitende Funktionen

- Zeitabstraktion führt auf stromverarbeitende Funktionen
- Wir können auch Algorithmen durch Funktionen auf Strömen darstellen

Stromverarbeitende Funktionen (3)

fct rest: Stream $\alpha \rightarrow$ Stream α

fct sum = (s: Stream Nat) Stream Nat:
first(s) & sum((first(s) + first(rest(s))) & rest(rest(s)))

fct sumk = (n: Nat, s: Stream Nat) Stream Nat:
n & sumk(n+ first(s), rest(s))

sum(s) = sumk(first(s), rest(s))

Stromverarbeitende Funktionen (4)

fct sadd = (s1, s2: Stream Nat) Stream Nat:
(first(s1) + first(s2)) & sadd(rest(s1), rest(s2))

fct smult = (s1, s2: Stream Nat) Stream Nat:
(first(s1) ∗ first(s2)) & smult(rest(s1), rest(s2))

fct enum = (n: Nat) Stream Nat: n & enum(n+1)

Stromverarbeitende Funktionen (5)

fct diff = Stream Int → Stream Int

diff(x & y & s) = (y − x) & diff(y & s)

fct integ = Int, Stream Int → Stream Int

integ(z, x & s) = z & integ(z + x, s)

integ(x, diff(x & y & s))
= integ(x, (y − x) & diff(y & s))
= x & integ(y, diff(y & s))

integ(first(s), diff(s)) = s

Stromverarbeitende Funktionen (6)

fct store = (c: Stream Bool, s: Stream Data) Stream Data:

store(true & c, s) = first(s) & store(c, s)
store(false & c, s) = store(c, rest(s))

Beispiel: Das Sieb des Eratosthenes

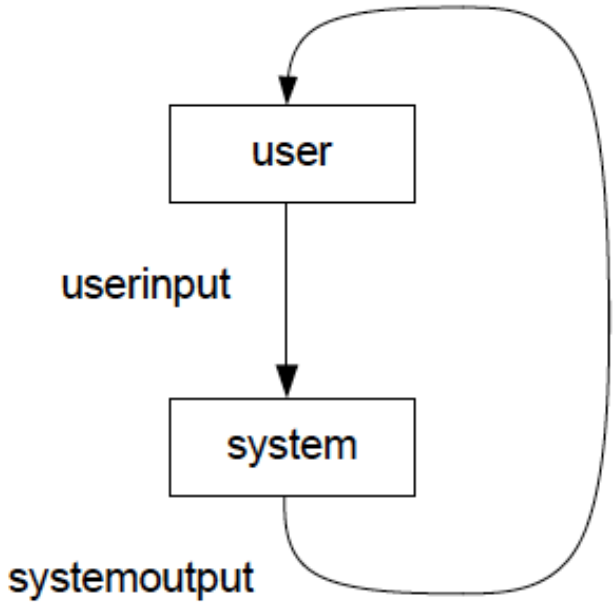
Unendlicher Strom aller Primzahlen:

```
Stream Nat erastosthenes = sieb(enum(2))

fct sieb = (s: Stream Nat) Stream Nat:
    first(s) & sieb(elim(first(s), s))

fct elim = (n: Nat, s: Stream Nat) Stream Nat:
  if n divides first(s) then elim(n, rest(s))
                        else first(s) & elim(n, rest(s))
  fi
```

Beispiel: Nutzer und System (1)

```
fct user = (Stream Output s, t: userstate) Stream Input:
   if enough(first(s), t) then ⟨⟩
                          else newinput(first(s), t ) &
                               user(rest(s), newuserstate(first(s), t))
   fi
fct system = ( Stream Input s, t: Systemstate) Stream Output:
   if ready(first(s), t)  then ⟨⟩
                          else newoutput(first(s), t) &
                               system(rest(s), newsystemstate(first(s), t))
   fi

Stream Output systemoutput = system(userinput, initialsystemstate)
Stream Input userinput = initialinput & user(systemoutput, initialuserstate)
```

Beispiel: Nutzer und System (2)



Abschließende Bemerkungen

- Schnittstellenverhalten kann durch die Interaktionsgeschichten (Austausch der Nachrichten über die Zeit) erfasst werden
- Kanäle und Ströme sind ein Konzept für die Beschreibung von Schnittstellen
- Spezielle Zustandsmaschinen arbeiten auf Strömen
- Auch Algorithmen können über Ströme formuliert werden
- Später behandeln wir
  - Komposition von Systemen über die Verbindung von Kanälen

Präfixordnung auf Strömen

Präfixordnung auf Strömen $\sqsubseteq \in M^{\infty} \times M^{\infty}$

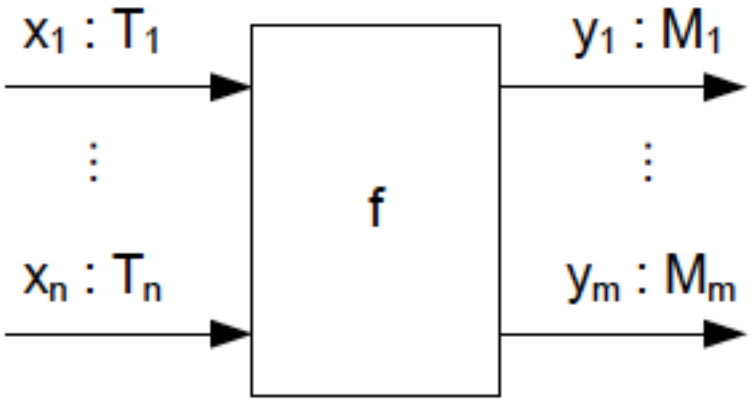
Definiert durch

$$x \sqsubseteq y \Leftrightarrow \exists\, z \in M^{\infty} : x \hat{} z = y$$

Die Relation ist eine partielle Ordnung mit kleinstem Element $\langle\rangle$

Stromverarbeitende Funktionen (2)

f : Stream $T_1 \times ... \times$ Stream $T_n \rightarrow$ Stream $M_1 \times ... \times$ Stream M_m



Monotonie:

$$
\begin{aligned}
& (y_1, \ldots, y_m) = f(x_1, \ldots, x_n) \\
\wedge\ & (y'_1, \ldots, y'_m) = f(x'_1, \ldots, x'_n) \\
\wedge\ & x_1 \sqsubseteq x'_1 \wedge \ldots \wedge x_n \sqsubseteq x'_n \\
\Rightarrow\ & y_1 \sqsubseteq y'_1 \wedge \ldots \wedge y_m \sqsubseteq y'_m
\end{aligned}
$$