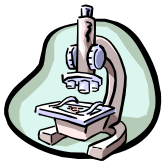
FOCUS

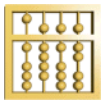
* * *



Modulare hierarchische Modellierung verteilter interaktiver nebenläufiger Systeme im Software & Systementwicklungsprozess

Manfred Broy

Technische Universität München
Institut für Informatik
D-80290 München, Germany

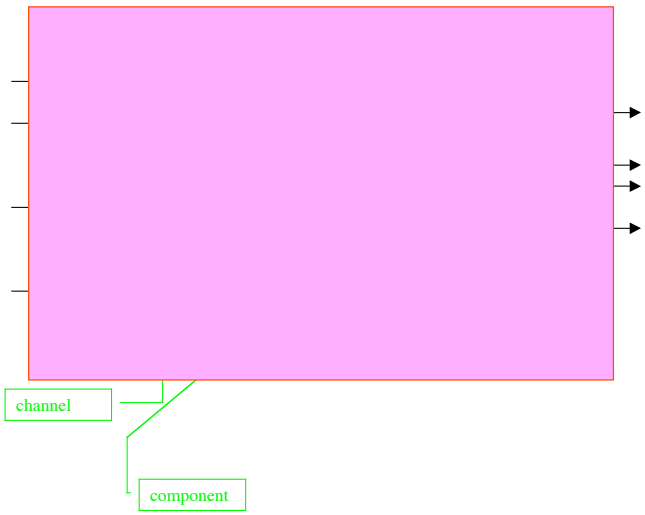


Hauptanliegen

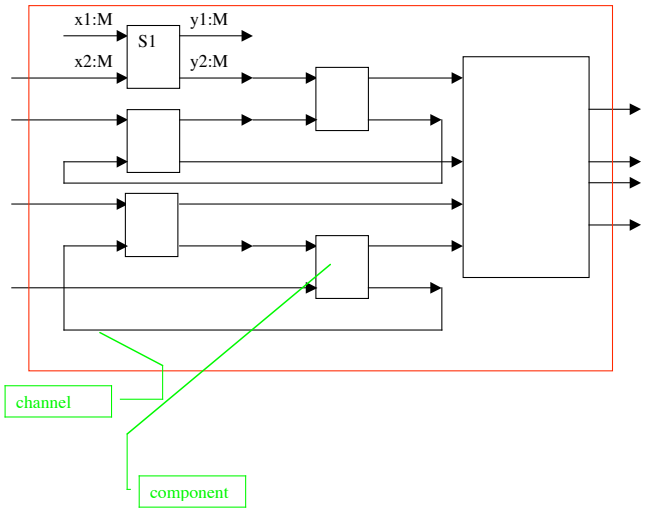
Theorie für die Modellierung verteilter interaktiver nebenläufiger Systeme unter Betonung von

- Schnittstellenabstraktion:
  Schnittstellenverhalten: Modellierung/Spezifikation aller Eigenschaften eines Systems, die für die Nutzung in beliebigen Kontexten von Belang sind.
- Kompositionalität:
  Aus den Schnittstellen der Teilsysteme (Komponenten) eines Systems können wir durch (parallele) Komposition die Schnittstellenabstraktion des Gesamtsystems gewinnen
- Verfeinerung/hierarchische Dekomposition:
  Schrittweise Entwicklung von Systemen in Hinblick auf Spezifikation, Architektur, Implementierung

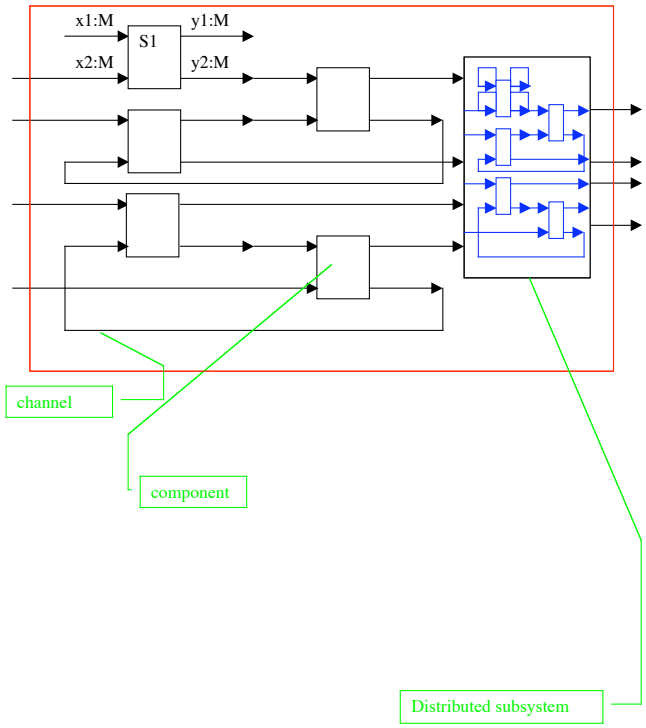
Architektur eines Systems: Hierarchische Verfeinerung



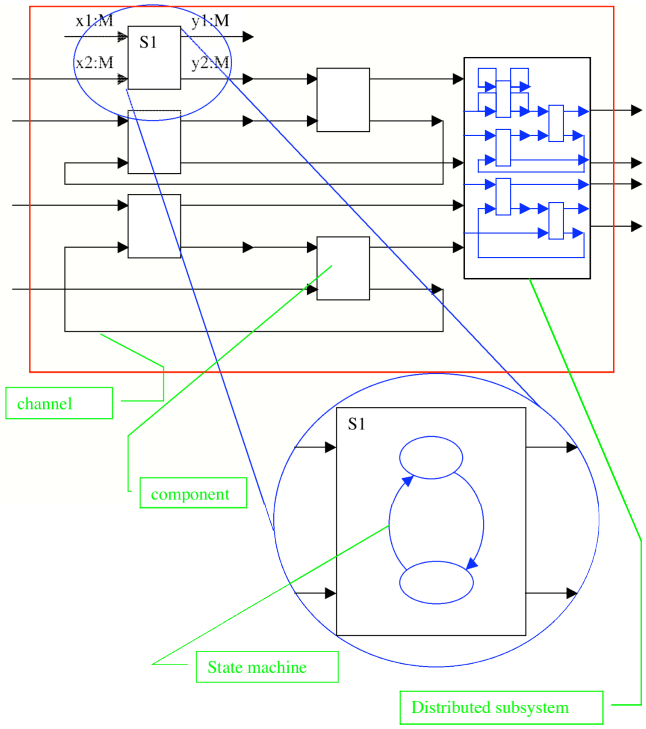
Architektur eines Systems: Hierarchische Verfeinerung



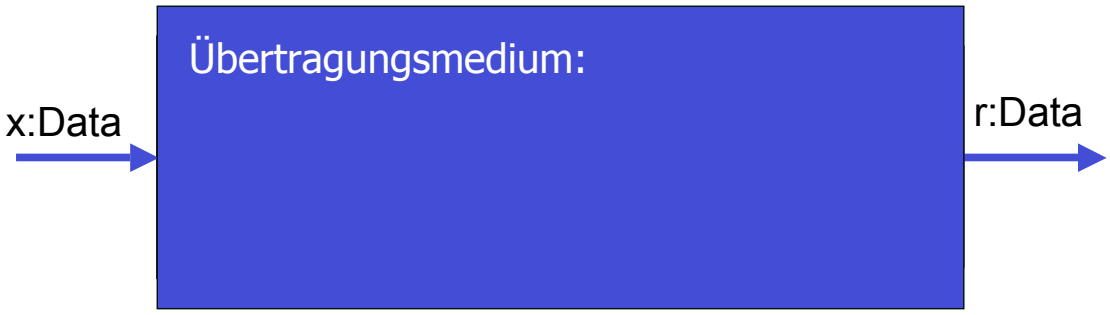
Architektur eines Systems: Hierarchische Verfeinerung



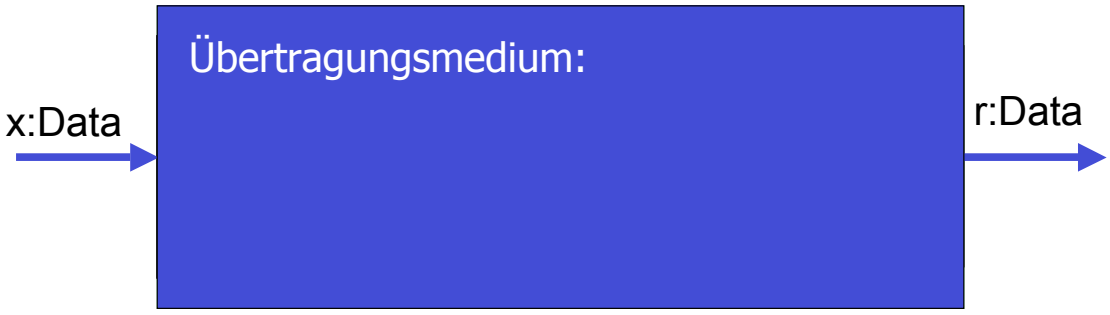
Architektur eines Systems: Hierarchische Verfeinerung



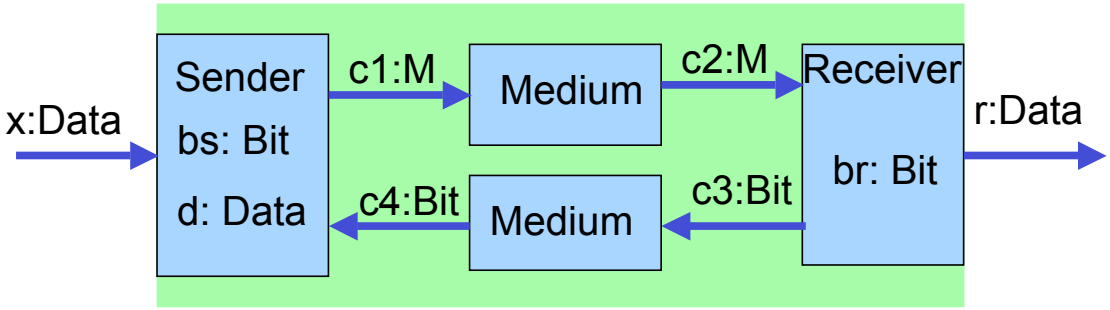
Ein motivierendes Beispiel: Alternating Bit Protokoll



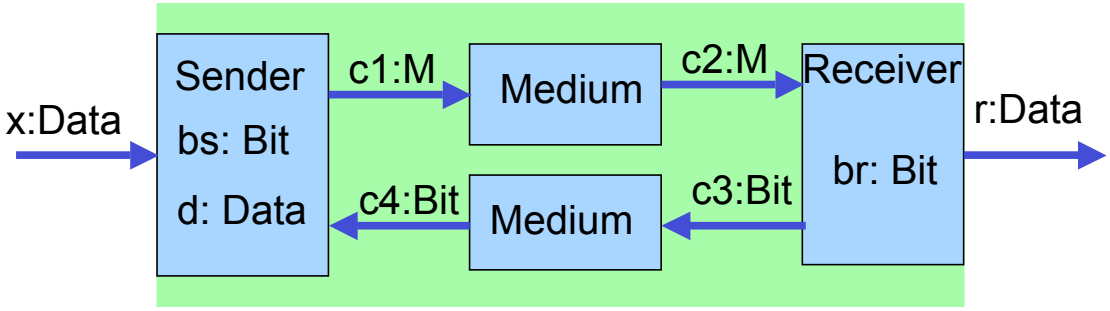
Ein motivierendes Beispiel: Alternating Bit Protokoll



Ein motivierendes Beispiel: Alternating Bit Protokoll



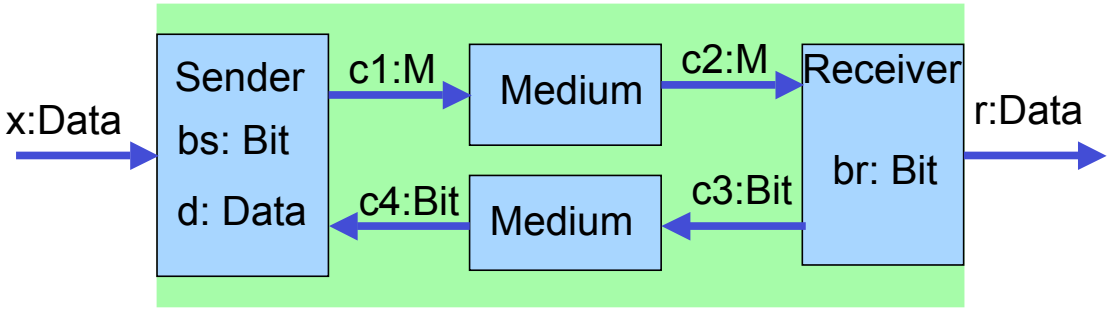
Ein motivierendes Beispiel: Alternating Bit Protokoll



Ergibt die Gleichungen:

c1 = sender(x, c4)
c2 = medium(c1)
(r, c3) = receiver(c2)
c4 = medium(c3)

Ein motivierendes Beispiel: Alternating Bit Protokoll



Ergibt die Gleichungen:

c1 = sender(x, c4)
c2 = medium(c1)
(r, c3) = receiver(c2)
c4 = medium(c3)

Im Falle von Nichtdeterminismus

c1 ∈ sender(x, c4)
c2 ∈ medium(c1)
(r, c3) ∈ receiver(c2)
c4 ∈ medium(c3)

Spezifikation des Alternating Bit Protokolls: Sender

Wir definieren eine „DeRepeater"-Relation:

derep: Stream M → Stream M

spezifiziert durch

derep(⟨⟩) = {⟨⟩}
derep(⟨m⟩^⟨m'⟩^x) = ⟨m⟩^derep(⟨m'⟩^x) ⇐ m ≠ m'
derep(⟨m⟩^⟨m⟩^x) = derep(⟨m⟩^x)
derep(⟨m⟩^∞) = ⟨m⟩

Damit spezifizieren wir

sender(x, c4) = {c1: ∃ z: derep(c1)^z = addbit(x, L)
∧ ((z = ⟨⟩ ∧ #derep(⟨O⟩^c4) = 1+#x)
∨ (#derep(⟨O⟩^c4) = #derep(c1)))}

wobei

addbit(⟨⟩, b) = ⟨⟩
addbit(⟨m⟩^x, b) = ⟨(m, b)⟩^ addbit(x, ¬b)

altbit(b) = = ⟨b⟩^altbit(¬b)

Spezifikation des Alternating Bit Protokolls: Receiver

Wir definieren eine „Repeater"-Relation:

receiver : Stream (Data×Bit) → Stream Data × Stream Bit

spezifiziert durch

receiver(c2) = {(r, c3): r = p1(derep(c2)) ∧ c3 = p2(c2) }

wobei

p1(⟨⟩) = ⟨⟩
p1(⟨(m, b)⟩^x) = ⟨m⟩^p1(x)

p2(⟨⟩) = ⟨⟩
p2(⟨(m, b)⟩^x) = ⟨b⟩^p2(x)

Spezifikation des Alternating Bit Protokolls: Receiver

Wir definieren eine „Repeater"-Relation:

receiver : Stream (Data×Bit) → Stream Data × Stream Bit

spezifiziert durch

receiver(c2) = {(r, c3): r = p1(derep(c2)) ∧ c3 = p2(c2) }

Spezifikationstableau

Receiver

in c2: (Data, Bit)
out r: Data, c3: Bit
r = p1(derep(c2))
∧ c3 = p2(c2)

Spezifikation des Alternating Bit Protokolls: Medium

Wir spezifizieren das Medium

$$\text{medium}(\langle\rangle) = \langle\rangle$$
$$\text{medium}(\langle b\rangle \hat{} c) = \text{medium}(c) \cup \langle b\rangle \hat{} \text{medium}(c)$$

wobei Fairness gelte

$$\#c = \infty \Rightarrow \#\text{medium}(c) = \infty$$

Ausgangspunkt

Fülle von Systembeschreibungsansätzen:

- Graphische Beschreibungstechniken (SDL, UML etc.)
- Prozeßalgebren (CSP, CCS, ...)
- Automaten (I/O-Automaten, Petri-Netze)
- Programmiersprachen
- Logik (Temporal, Modal etc., TLA, Unity, ...)
- E/R-Modelle
- Relationen

Beobachtungen zu graphischen Beschreibungstechniken

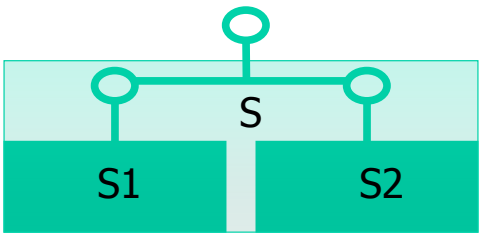
Am Beispiel UML:

- Semantik oft unklar
- Modelle isoliert betrachtet
- Sichten nicht integriert
- Bestimmte Aspekte schlecht beherrscht (Mobilität, Zeit)
- Modelle nicht adäquat
- Verfeinerung/Simulation nicht wohldefiniert
- Hardware/Software-Coop-Modelle fehlen
- Fehlende Schnittstellenverhaltensbeschreibung
- Architekturbeschreibungen unpräzise
- Keine hierarchische Zerlegung

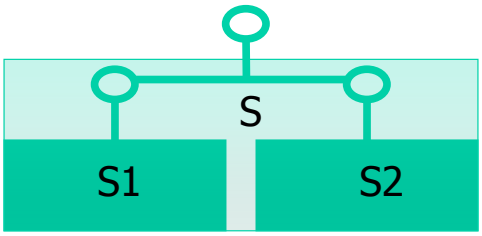
Ziele - Focus

- Schnittstellensicht - Schnittstellenspezifikation
- Modularität
- Mult-Sichtenmodell
- Sichten integrieren
- Systemmodelle vervollständigen
- Theorien schaffen
- Schichtenansatz
- Abbildung auf Logik
- Grundlage für Engineering
- Ein-/Ausgabe, Ursache/Wirkungsbeziehung, Kausalität
- Nichtdeterminismus/Unterspezifikation

Komposition / Dekomposition von Systemen

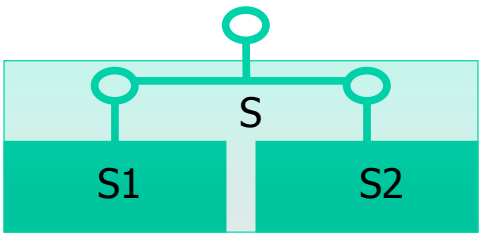


Komposition / Dekomposition von Systemen



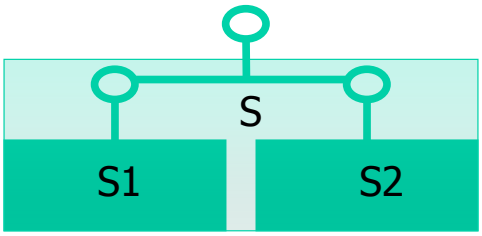
Komposition $S = S1 \otimes S2$

Komposition / Dekomposition von Systemen



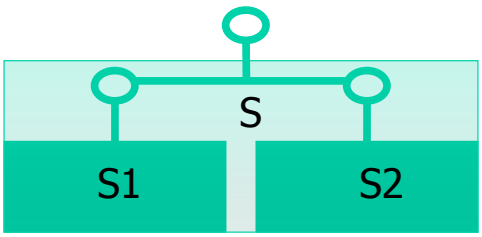
Komposition	$S = S1 \otimes S2$
Abstraktion	α : System → Verhalten

Komposition / Dekomposition von Systemen



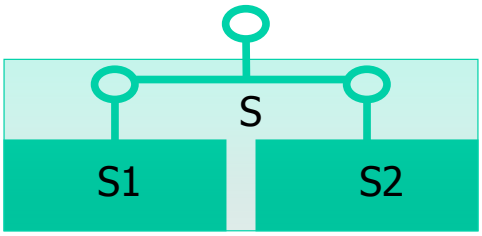
Komposition	$S = S1 \otimes S2$
Abstraktion	α : System → Verhalten
Beobachtung:	β : System → Beobachtung

Komposition / Dekomposition von Systemen



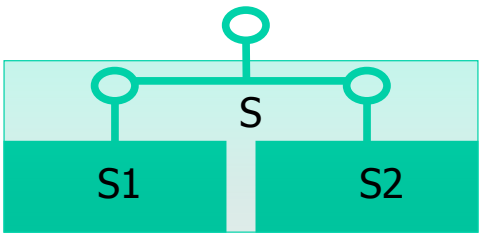
Komposition	$S = S1 \otimes S2$
Abstraktion	α : System → Verhalten
Beobachtung:	β : System → Beobachtung
Kompositionalität:	$\alpha(S) = \alpha(S1) \otimes' \alpha(S2)$

Komposition / Dekomposition von Systemen



Komposition	$S = S1 \otimes S2$
Abstraktion	α : System → Verhalten
Beobachtung:	β : System → Beobachtung
Kompositionalität:	$\alpha(S) = \alpha(S1) \otimes' \alpha(S2)$
Ausdrucksstärke:	γ : Verhalten → Beobachtung

Komposition / Dekomposition von Systemen



Komposition	$S = S1 \otimes S2$
Abstraktion	α : System → Verhalten
Beobachtung:	β : System → Beobachtung
Kompositionalität:	$\alpha(S) = \alpha(S1) \otimes' \alpha(S2)$
Ausdrucksstärke:	γ : Verhalten → Beobachtung
	$\gamma(\alpha(S)) = \beta(s)$

Themen

- Theorie
  - Ströme
  - Komponenten
  - Zeit
  - Komposition
- Erweiterungen
- Zustandsmaschinen
- Methodische Aspekte
  - Spezifikation
  - Verifikation
  - Verfeinerung
  - Theorie
  - Algebra
- Anwendungen
  - Semantikdefinition
    - MSCS
    - UML
    - SDL
  - Werkzeuge
    - Autofocus

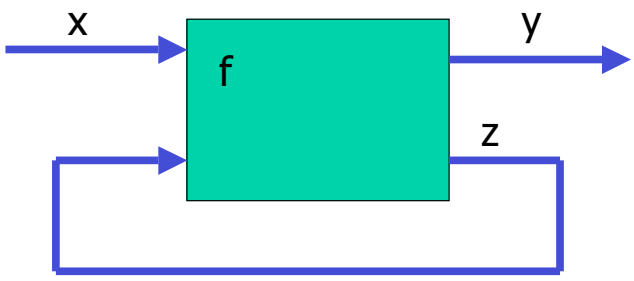
Modellierung der Kanäle: Ströme

Terminologie

- Ein Kanal verbindet zwei Teilsysteme oder ein system mit seiner Umgebung (Ein- oder Ausgabekanal)
- Ein Strom Modellierung die Kommunikationsgeschichte auf einem Kanal

  Da in zusammengesetzten Systemen Ströme durch rekursive Gleichungen definiert werden, brauchen wir eine Theorie der Ströme, die rekursive Gleichungen unterstützt.

Elementares Beispiel: Rückkopplung



Entspricht der Gleichung:

(y, z) = f(x, z)

Problem:

- Wie sicherstellen, dass Gleichung Lösung hat
- Wie Lösung wählen, wenn mehre Lösungen ex.

Ströme

Sei M eine Menge von Nachrichten

M^* Menge der endliche Sequenzen über M

$\langle\rangle$ leere Sequenz

M^∞ Menge der unendlichen Sequenzen über M

$\mathbb{N} \setminus \{0\} \rightarrow M$

M^ω Menge der Menge der Ströme

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

Konzepte und Gesetze auf Strömen

$x^\frown y$	Konkatenation der Ströme x, y
$\langle\rangle^\frown x = \langle\rangle = x^\frown\langle\rangle$	$\langle\rangle$ is neutrales Element
$x^\frown(y^\frown z) = (x^\frown y)^\frown z$	Assoziativität
$x^\frown y = x$	falls x unendlich ist
Präfixordnung:	$x \sqsubseteq y \equiv \exists\, z \in M^\omega: x^\frown z = y$
$(M^\omega, \sqsubseteq)$	ist partiell geordnete Menge mit $\langle\rangle$ als kleinstes Element, kettenvollständig

Konsequenz

Jede monotone Funktion über Strömen hat kleinsten Fixpunkt

Montonie

$x \sqsubseteq z \Rightarrow F.x \sqsubseteq F.z$

Dies erfüllt unsere Zielsetzung für deterministische Systeme vollständig!

Problem: Verallgemeinerung auf nichtdeterministische Systeme

Lösungsansatz: Gezeitete Ströme, Kausalität

Gezeiteter Strom:

$x : IN\backslash\{0\} \rightarrow M^*$

Menge der gezeiteten Ströme $(M^*)^\infty$

Systemmodel

Grundlegendes Model

Gezeitete Ströme: Semantisches Model für Schnittstellenverhalten

Nachrichtenmenge:

M = {a, b, c, ...}

Kanalbelegungen

Kanal: Typisierter Identifikator für einen Strom

Sei C Menge von Kanälen

Kanalbelegung für C:

$x: C \rightarrow (M^*)^\infty$ korrekt typisiert

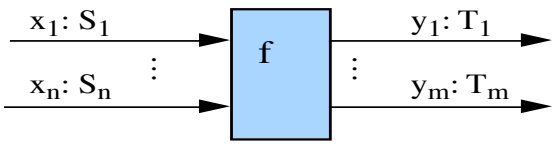
$\vec{C}$ Menge aller Kanalbelegungen

Interface model

$I = \{ x_1 , x_2 , ... \}$ Menge der Eingabekanäle
$O = \{ y_1 , y_2 , ... \}$ Menge der Ausgabekanäle

Schnittstellenverhalten

$$f : \vec{I} \rightarrow \wp(\vec{O})$$



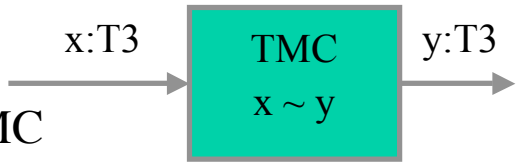
Example: Interface specification

A transmission component TMC

TMC
in x: T3 **out** y: T3
x ~ y

$x \sim y \equiv (\forall\, m \in T3: \{m\} \copyright \overline{x} = \{m\} \copyright \overline{y})$

Example: Interface specification

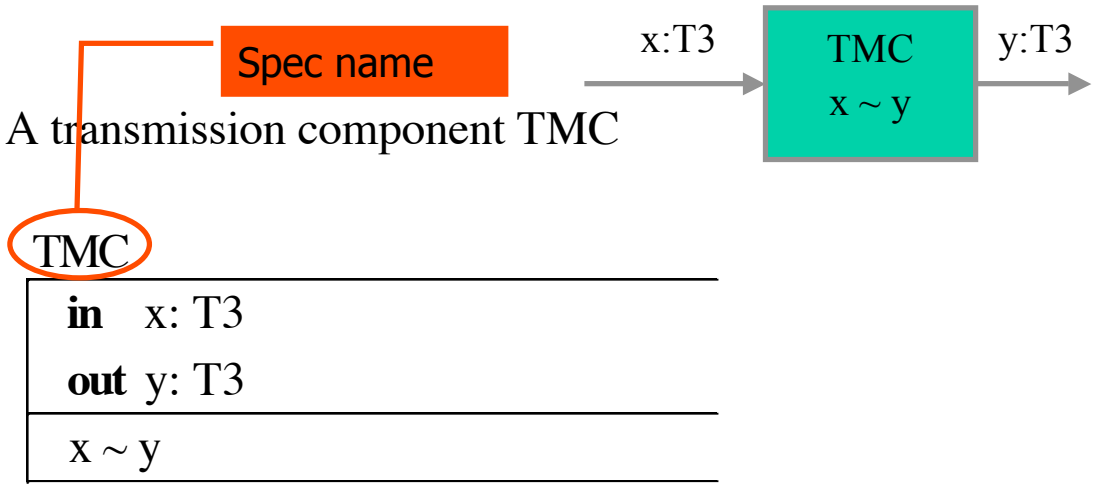


A transmission component TMC

TMC
in x: T3 **out** y: T3
x ~ y

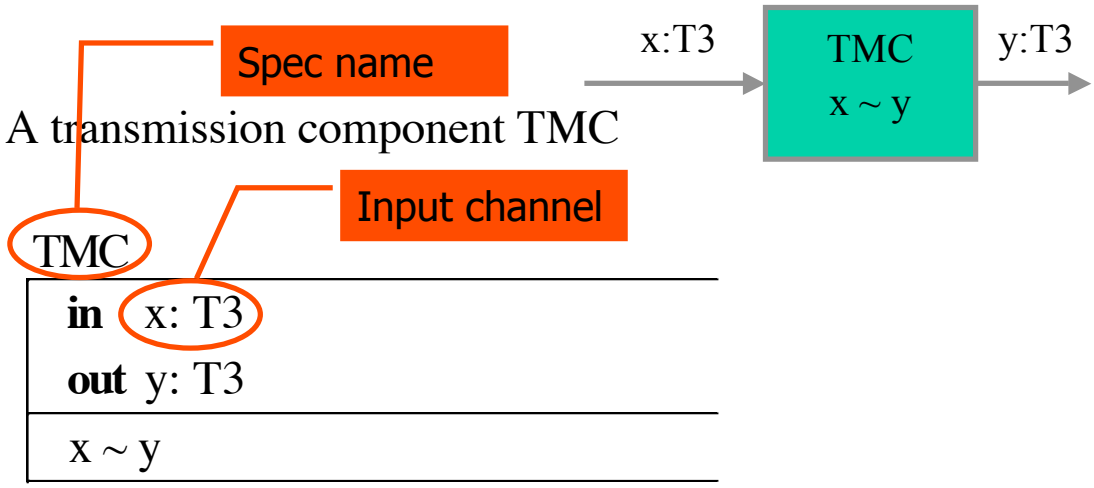
$x \sim y \equiv (\forall\, m \in T3: \{m\} \copyright \overline{x} = \{m\} \copyright \overline{y})$

Example: Interface specification



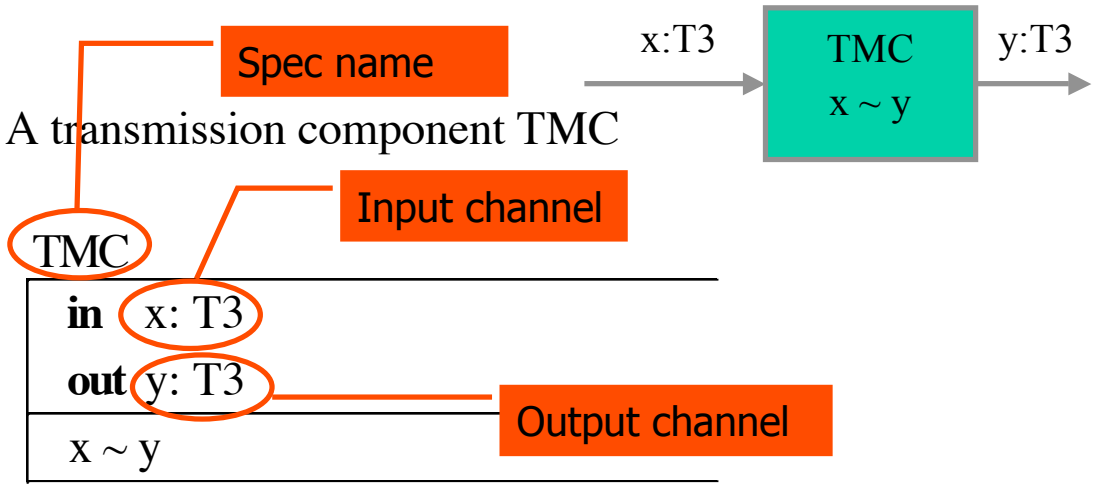
$x \sim y \equiv (\forall\, m \in T3: \{m\} \copyright \overline{x} = \{m\} \copyright \overline{y})$

Example: Interface specification



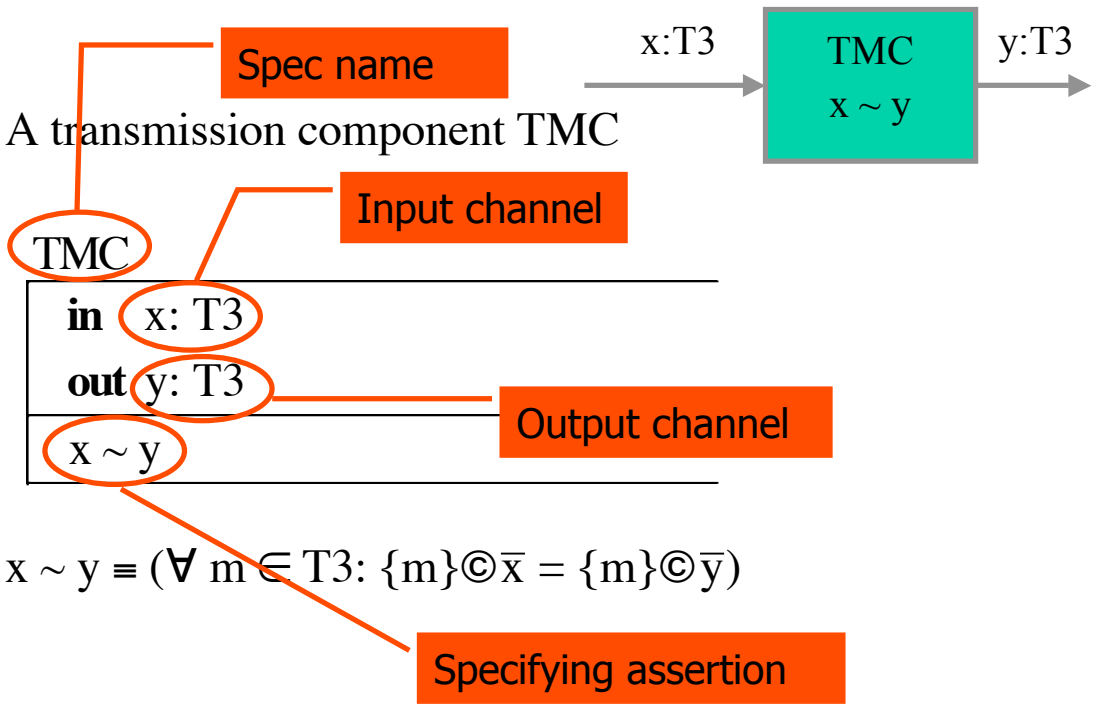
$x \sim y \equiv (\forall\, m \in T3: \{m\} \copyright \overline{x} = \{m\} \copyright \overline{y})$

Example: Interface specification



$x \sim y \equiv (\forall\, m \in T3: \{m\}©\overline{x} = \{m\}©\overline{y})$

Example: Interface specification



$x \sim y \equiv (\forall\, m \in T3: \{m\}©\overline{x} = \{m\}©\overline{y})$

Specifying assertion

Komponentenschnittstellen und Kausalität

$F : \vec{I} \rightarrow \wp(\vec{O})$ *Schnittstellverhalten*

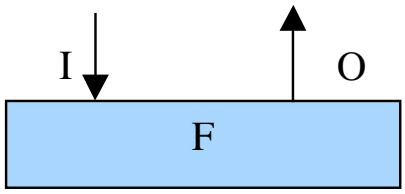
Kausalität

(let $x, z \in \vec{I}$, $y \in \vec{O}$, $t \in IN$):

$$x{\downarrow}t = z{\downarrow}t \Rightarrow \{y{\downarrow}t+1 : y \in F(x)\} = \{y{\downarrow}t+1 : y \in F(z)\}$$

$x{\downarrow}t$ Präfix der ersten t Zeitintervalle

Eine kausale Komponente F ist total, d.h. $F.x \neq \varnothing$ für alle x, oder $F.x = \varnothing$ für alle x



Konsequenzen der Kausaliät

- Korrekte Abstraktion einer operationellen Semantik (Zustandsmaschine)
- Zeitfluss wird adäquat im Modell erfasst
- Auf deterministischen Systemen sind Fixpunkte eindeutig
- Kausalität kann schematisch zu Spezifikationen hinzugenommen werden
- Kausalität induziert Induktionsprinzip

Zustandsmaschinen

Σ Zustandsraum: Menge von Zuständen

Zustandsübergangsfunktion:

$$\Delta: (\Sigma \times (I \rightarrow M^*)) \rightarrow \wp(\Sigma \times (O \rightarrow M^*))$$

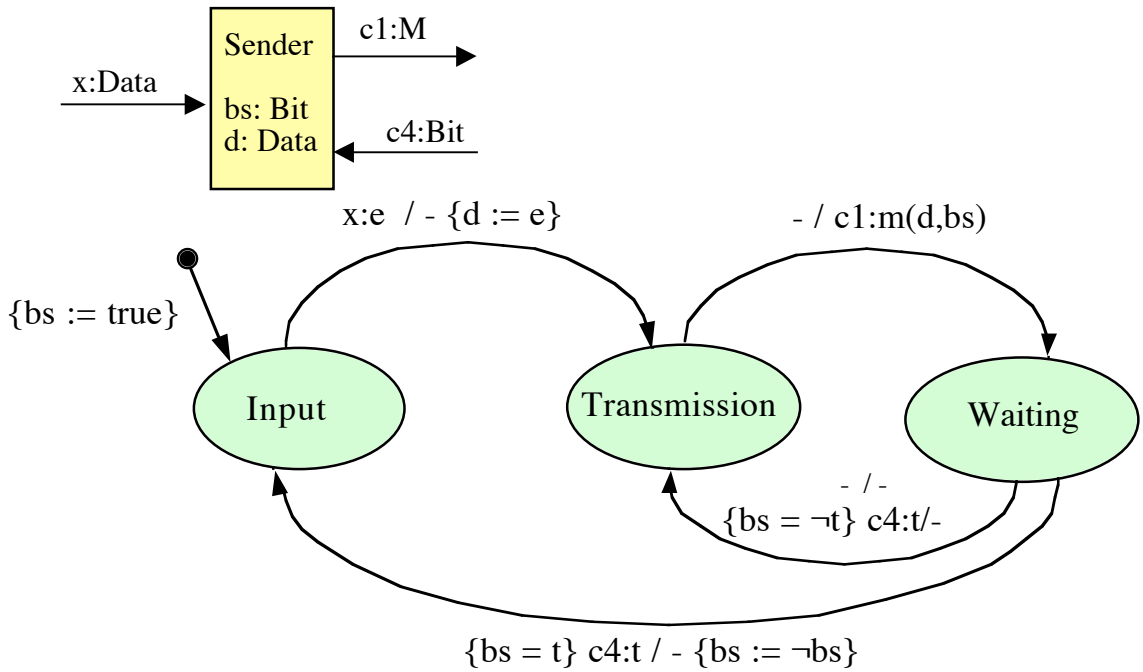
Zustandsmaschine:

$$(\Delta, \Lambda)$$

Menge der Anfangszustände:

$$\Lambda \subseteq \Sigma \times (O \rightarrow M^*)$$

Zustandsübergangsdiagramm



Schnittstellenabstraktion für Zustandsmaschinen

Die Zustandsübergangsfunktion Δ induziert eine Funktion

$$B_\Delta : \Sigma \to ((O \to M^*) \to (\vec{I} \to P(\vec{O})))$$

Für jeden Zustand $\sigma \in \Sigma$,

jede initiale Ausgabe $y_0 \in (O \to M^*)$,

jedes Eingabemuster $z \in (I \to M^*)$, und

jede Belegung x der Eingabekanäle gelte

$B_\Delta(\sigma, y_0).(\langle z \rangle \hat{} x) =$

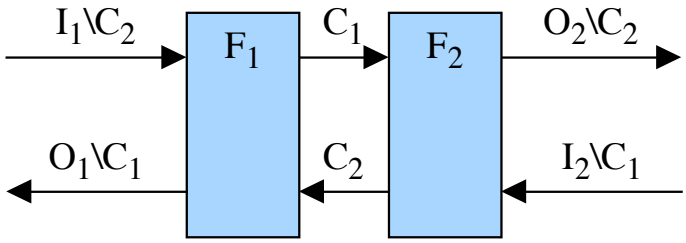
$\{\langle y_0 \rangle \hat{} y: \exists\, \sigma' \in \Sigma, r \in (O \to M^*): (\sigma', r) \in \Delta(\sigma, z) \wedge y \in B_\Delta(\sigma', r).x\}$

B_Δ heißt Schnittstellenabstraktion

Komposition

Gegeben

$F_1 \in IF[I_1 \blacktriangleright O_1]$
$F_2 \in IF[I_2 \blacktriangleright O_2]$
$C_1 \subseteq O_1 \cap I_2$
$C_2 \subseteq O_2 \cap I_1$

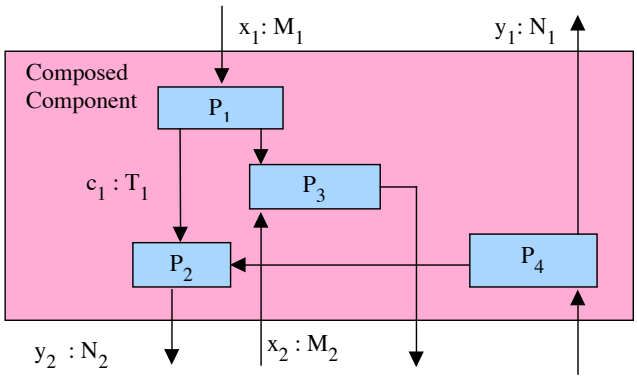


$F_1[C_1 \leftrightarrow C_2]F_2 \in IF[I_1 \backslash C_2 \cup I_2 \backslash C_1 \blacktriangleright O_1 \backslash C_1 \cup O_2 \backslash C_2]$,

$(F_1[C_1 \leftrightarrow C_2]F_2).x = \{z|(O_1 \backslash C_1) \cup (O_2 \backslash C_2): x = z|\, I_1 \backslash C_2 \cup I_2 \backslash C_1$
$\wedge\ z|O_1 \in F_1(z|I_1) \wedge z|O_2 \in F_1(z|I_2)\}$

Komposition von Spezifikationen

Komposition von Spezifikationen



Focus, Berlin Mai 2004 Manfred Broy 30

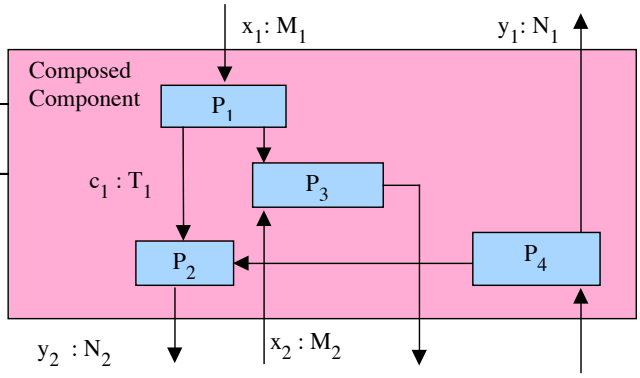
Komposition von Spezifikationen

Composed components spec

in $x_1: M_1 , x_2: M_2 , ...$

out $y_1: N_1 , y_2: N_2 , ...$

$\exists c_1, c_2 , ... : P_1 \wedge ... \wedge P_n$



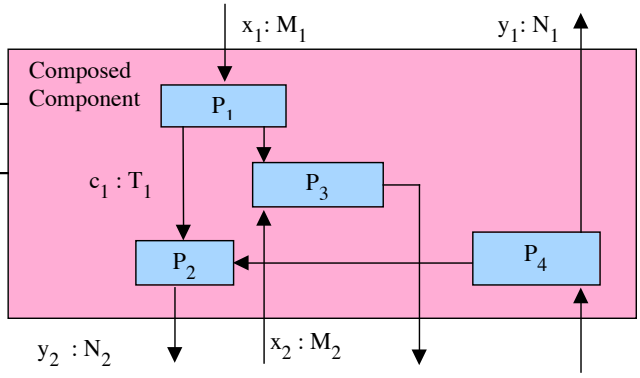
Komposition von Spezifikationen

Composed components spec

in $x_1: M_1 , x_2: M_2 , ...$

out $y_1: N_1 , y_2: N_2 , ...$

$\exists c_1, c_2 , ... : P_1 \wedge ... \wedge P_n$



Systemkomposition = logisches UND

Komposition von Spezifikationen

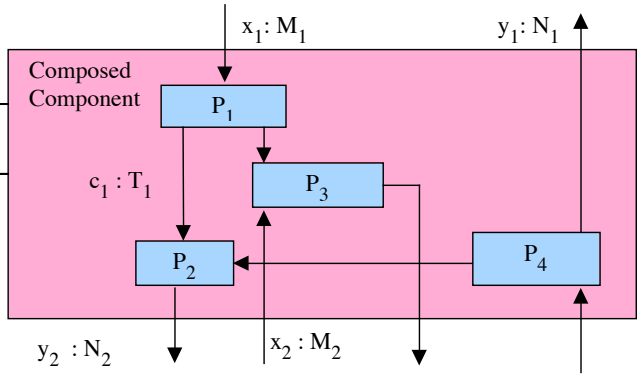
Eingabekanäle

Composed components spec

in $x_1: M_1 , x_2: M_2 , ...$

out $y_1: N_1 , y_2: N_2 , ...$

$\exists c_1, c_2 , ... : P_1 \wedge ... \wedge P_n$



Systemkomposition = logisches UND

Komposition von Spezifikationen

Eingabekanäle

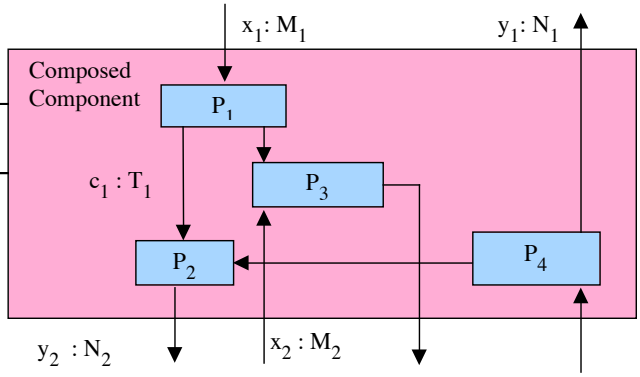
Ausgabekanäle

Composed components spec

in $x_1: M_1 , x_2: M_2 , ...$

out $y_1: N_1 , y_2: N_2 , ...$

$\exists c_1, c_2 , ... : P_1 \wedge ... \wedge P_n$



Systemkomposition = logisches UND

Komposition von Spezifikationen

Eingabekanäle

Ausgabekanäle

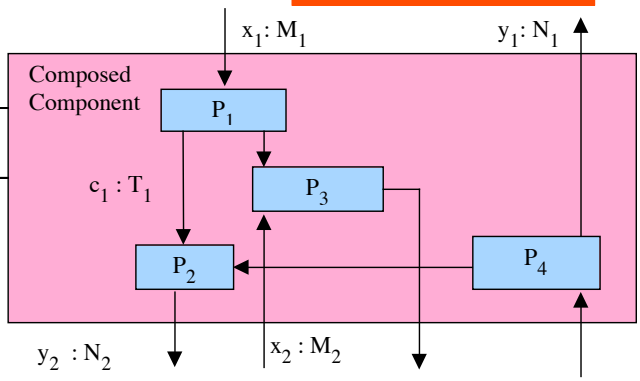
Composed components spec

in $x_1: M_1 , x_2: M_2 , ...$

out $y_1: N_1 , y_2: N_2 , ...$

$\exists c_1, c_2 , ... : P_1 \wedge ... \wedge P_n$

Interne Kanäle



Systemkomposition = logisches UND

Komposition von Spezifikationen

Eingabekanäle

Ausgabekanäle

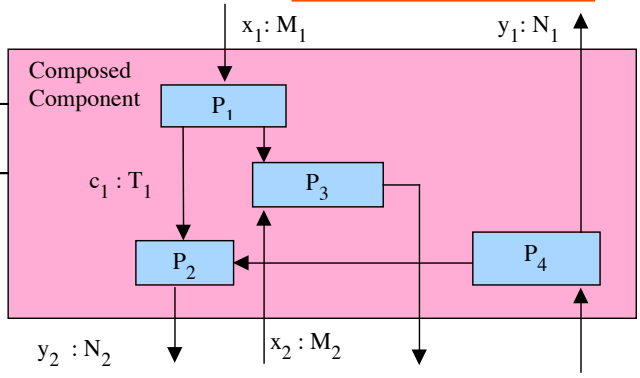
Composed components spec

in $x_1: M_1 , x_2: M_2 , ...$

out $y_1: N_1 , y_2: N_2 , ...$

$\exists c_1, c_2 , ... : P_1 \wedge ... \wedge P_n$

Interne Kanäle



Systemkomposition = logisches UND

Channel Hiding = Existenzquantifikation

Eigenschaftsverfeinerung

Eigenschaftsverfeinerung

$$F: \vec{I} \to \wp(\vec{O})$$

wird durch folgendes Verhalten verfeinert

$$\hat{F}: \vec{I} \to \wp(\vec{O})$$

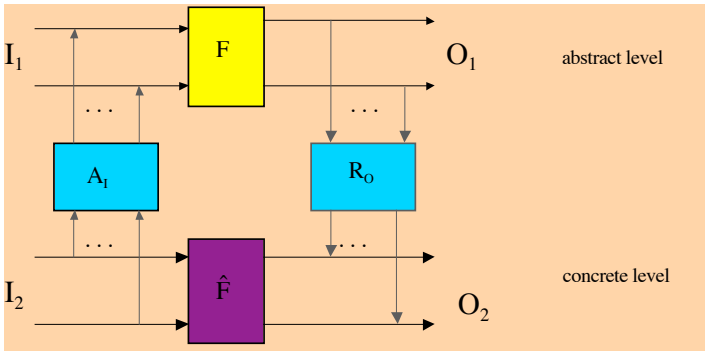
falls

$$\hat{F} \subseteq F$$

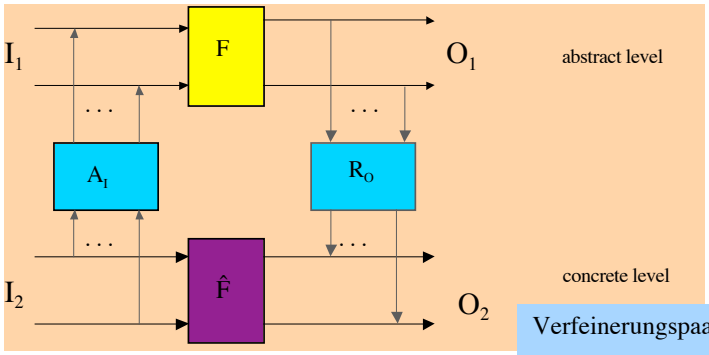
genauer

$$\forall\, x \in \vec{I}: \hat{F}.x \subseteq F.x$$

Verfeinerung über Abstraktionsebenen



Verfeinerung über Abstraktionsebenen



Verfeinerungspaare

$A_I: \vec{I}_2 \to \wp(\vec{I}_1)$ $R_I: \vec{I}_1 \to \wp(\vec{I}_2)$

$A_O: \vec{O}_2 \to \wp(\vec{O}_1)$ $R_O: \vec{O}_1 \to \wp(\vec{O}_2)$

stellen eine Beziehung zwischen Ein/Ausgabebelegungen her

$\hat{F}: \vec{I}_2 \to \wp(\vec{O}_2)$

heißt Interaktionsverfeinerung

$F: \vec{I}_1 \to \wp(\vec{O}_1)$

falls

$\hat{F} \subseteq A_I \circ F \circ R_O$ *U^1-simulation*

Ausblick

- Bezug zu graphischen Beschreibungsverfahren
- Theorie
  - Kompositionalität
  - Hierarchie
  - Verfeinerung
  - Spezifikation
  - Verifikation
- Engineering Notation
- Besondere Aspekte
  - Zeit
  - Mobilität
  - Instanzbildung
  - Dienste
  - Schichtenarchitekturen