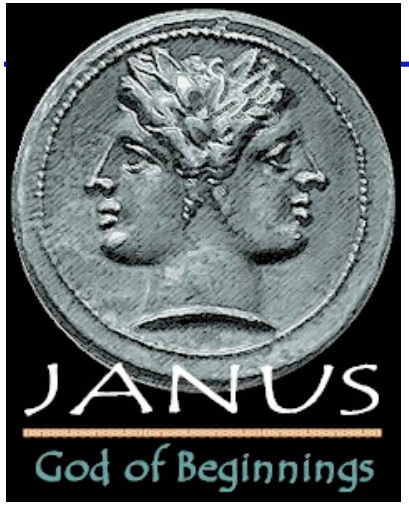


From Objects

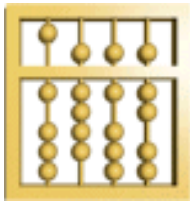
to Components

to Services

3.Lecture: Services

Manfred Broy

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



Components

$(I \blacktriangleright O)$ *syntactic interface* with set of input channels I and of output channels O

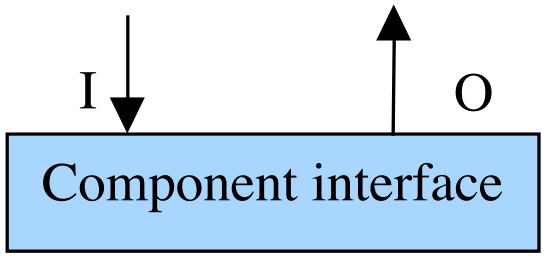
$F : \vec{I} \to \wp(\vec{O})$ component interface for $(I \blacktriangleright O)$ with *timing property* (let $x, z \in \vec{I}$, $y \in \vec{O}$, $t \in \mathbb{N}$):

Causality

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

$x{\downarrow}t$ prefix of x with t finite sequences

A component is a total behaviour



Causality

Consider the identity function:

$$F : \vec{C} \rightarrow \wp(\vec{C})$$

where $y \in F.x \Rightarrow \overline{x} = \overline{y}$

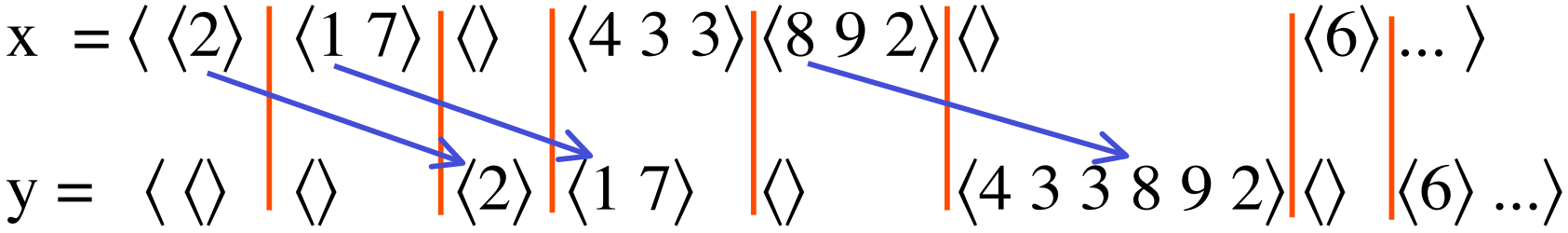
causality enforces

The golden rule of communication:

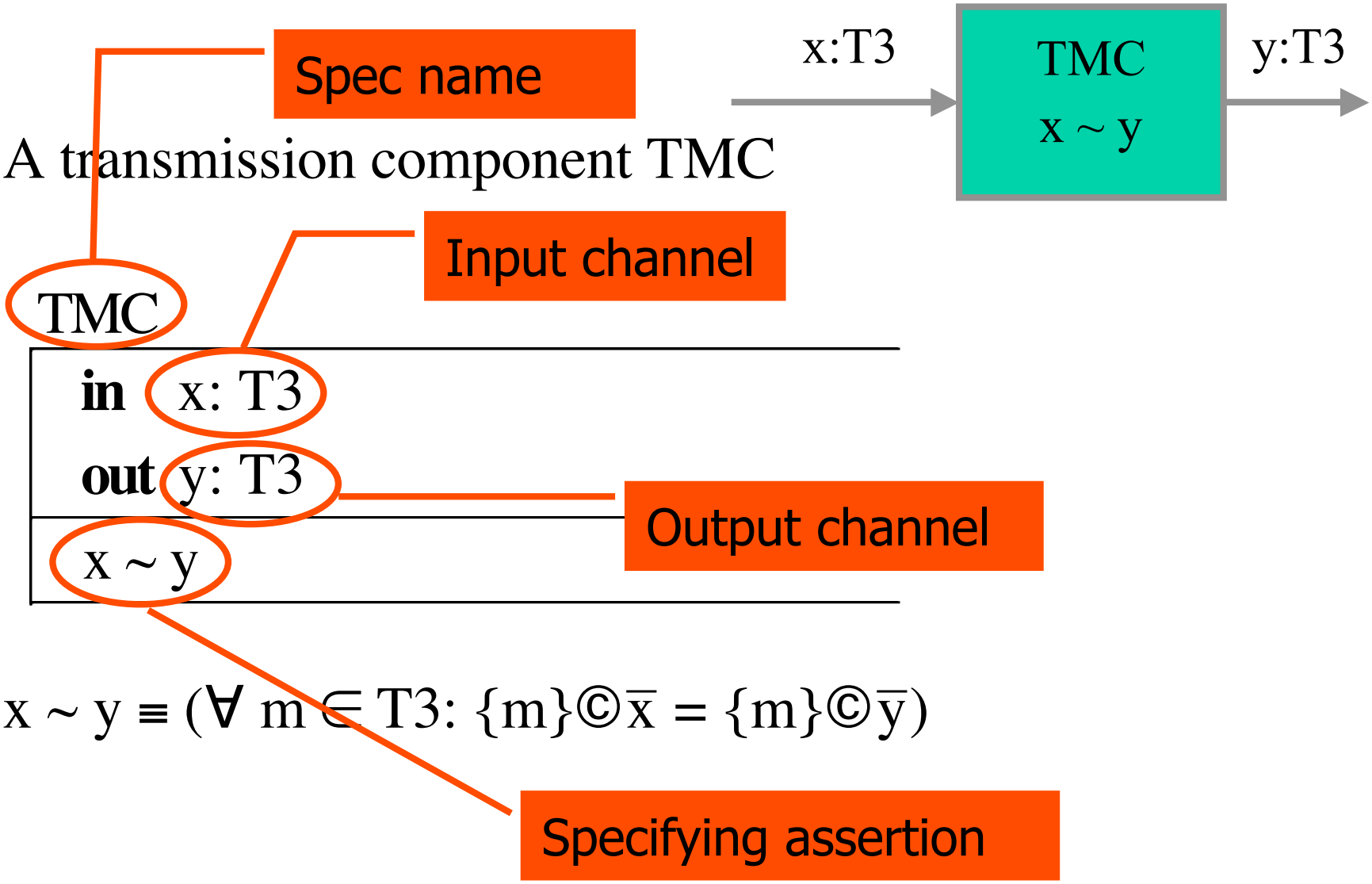
An information can never occur as output before it was received as input

$$F.x = \{y: \overline{x} = \overline{y} \wedge \forall\, t \in IN: \overline{y \downarrow t+1} \sqsubseteq \overline{x \downarrow t}\}$$

Example:

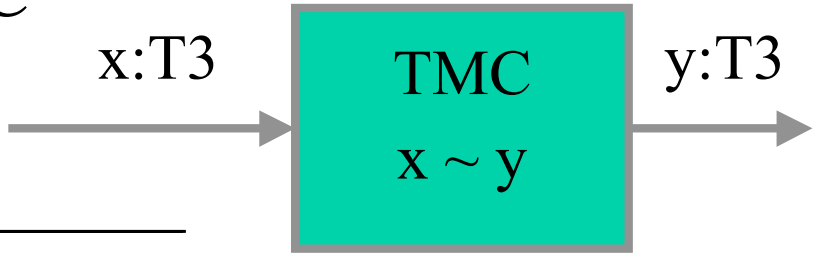


Interface specification: How to specify services



Interface specification: How to specify services

A transmission component TMC



TMC

in x: T3
out y: T3
x ~ y

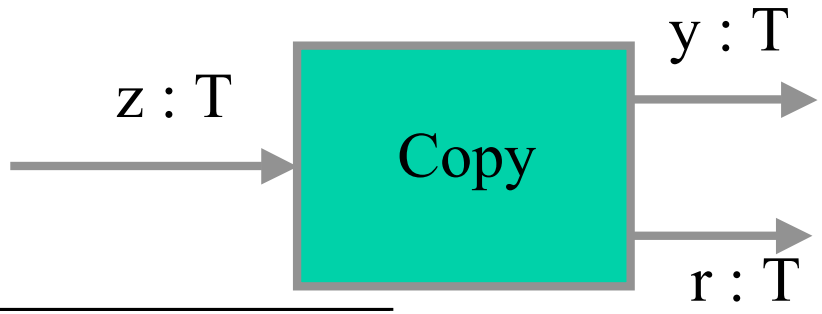
$$x \sim y \equiv (\forall\, m \in T3: \{m\}©\bar{x} = \{m\}©\bar{y})$$

Causality allows to conclude in addition

$$\forall\, t \in IN: \forall\, m \in T: \{m\}\#\overline{y \downarrow t+1} \leq \{m\}\#\overline{x \downarrow t}$$

Example: Interface specification

Component Copy



Copy

in z: T
out y, r: T
$\overline{z} = \overline{y} \wedge \overline{z} = \overline{r}$

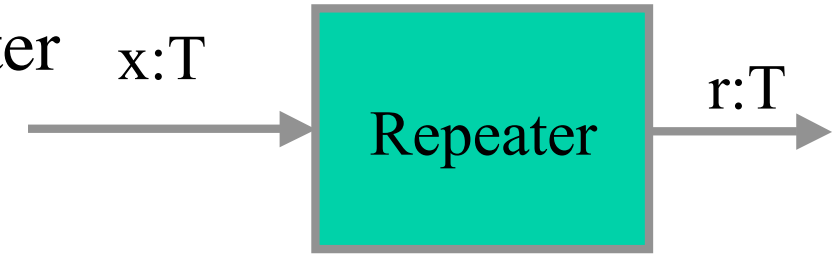
Causality allows to conclude in addition

$$\forall t \in IN: \forall m \in T:$$

$$\{m\}\#\overline{r \downarrow t+1} \leq \{m\}\#\overline{z \downarrow t} \wedge \{m\}\#\overline{y \downarrow t+1} \leq \{m\}\#\overline{z \downarrow t}$$

Example: Interface specification Repeater

Specification component Repeater



Repeater

in x: T

out r: T

$$\forall m \in T: \{m\}\#\overline{x} > 0 \Rightarrow \{m\}\#\overline{r} = \infty$$
$$\wedge \{m\}\#\overline{x} = 0 \Rightarrow \{m\}\#\overline{r} = 0$$

Causality allows to conclude in addition

$$\forall t \in \mathbb{N}:$$
$$\forall m \in T: \{m\}\#\overline{x \downarrow t} = 0 \Rightarrow \{m\}\#\overline{r \downarrow t+1} = 0$$

Remarks

- We get a very powerful, expressive model and a specification technique
- Nondeterminism/underspecification included
- The specification is done in pure higher order logic
- Verification calculus is therefore straightforward
- It is an attempt to eat the cake and keep it
  - ◇ Time is at your disposal whenever you need it but can be ignored whenever you like
  - ◇ Causality can be kept implicit, if not needed, but can be made explicit if more sophisticated reasoning is needed
  - ◇ Causality guarantees fixpoints in the case of networks of (deterministic) components

Consequences of causality

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

where $F.z = \varnothing$ for some $z \in \vec{I}$

then $F.x = \varnothing$ for all $x \in \vec{I}$

Proof: Causality

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

implies, since $x{\downarrow}0 = z{\downarrow}0$

$$\{y{\downarrow}1 : y \in F(x)\} = \quad \{y{\downarrow}1 : y \in F(z)\} = \quad \varnothing$$

Thus $F(x) = \varnothing$

Services

$F : \vec{I} \rightarrow \wp(\vec{O})$ service interface with the syntactic interface $(I \blacktriangleright O)$ with

$$F.x \neq \emptyset \neq F.z \wedge x{\downarrow}t = z{\downarrow}t \Rightarrow$$

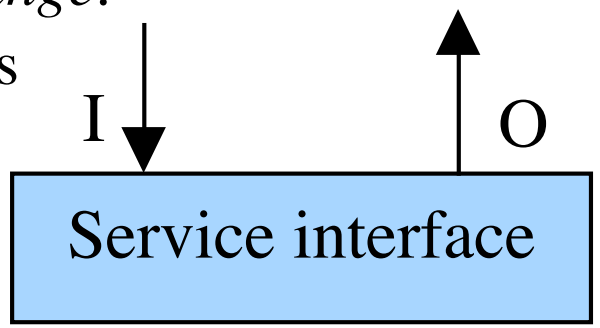
$$\{y{\downarrow}t+1 : y \in F(x)\} = \{y{\downarrow}t+1 : y \in F(z)\}$$

$Dom(F) = \{x : F.x \neq \emptyset\}$ the *service domain*

$Ran(F) = \{y \in F.x : x \in Dom(F)\}$ the *service range*.

$IF[I \blacktriangleright O]$ set of services

A service is a partial behaviour



Consequences of these definitions

- A component is a total behaviour!
- In contrast, a service is, in general, a partial behaviour!
- A component is the special case of a total service.
- Services can be related to components
  - ◇ A component may offer several services
- A multi-functional system/component can be defined in terms of a family of services that it offers

Example of a service: Queue

We specify the simple service of a queue

type QIn = {req} ∪ Data

type QOut = Data

Queue

in x: QIn **out** y: QOut
{req}#x = Data#y ∧ $\overline{y} \sqsubseteq$ Data©$\overline{x}$

The input assumption here is:

$$\forall\, x' : x' \sqsubseteq \overline{x} \Rightarrow \{req\}\#x \leq Data\#x$$

Example of a service: Box

We specify the simple service of a queue

type QIn = {req} ∪ Data

type QOut = Data

Box

in x: QIn **out** y: QOut
{req}#x = Data#y ∀ d ∈ Data: {d}#y ≤ {d}#x

The input assumption here is:

$$\forall\, x' : x' \sqsubseteq \overline{x} \;\Rightarrow\; \{req\}\#x' \leq Data\#x'$$

Example of a service: Unreliable Queue

We specify the simple service of an unreliable queue

type QIn = {req} ∪ Data

type QOut = {fail} ∪ Data

Here we have to refer explicitly to the output history in the input assumption

Queue

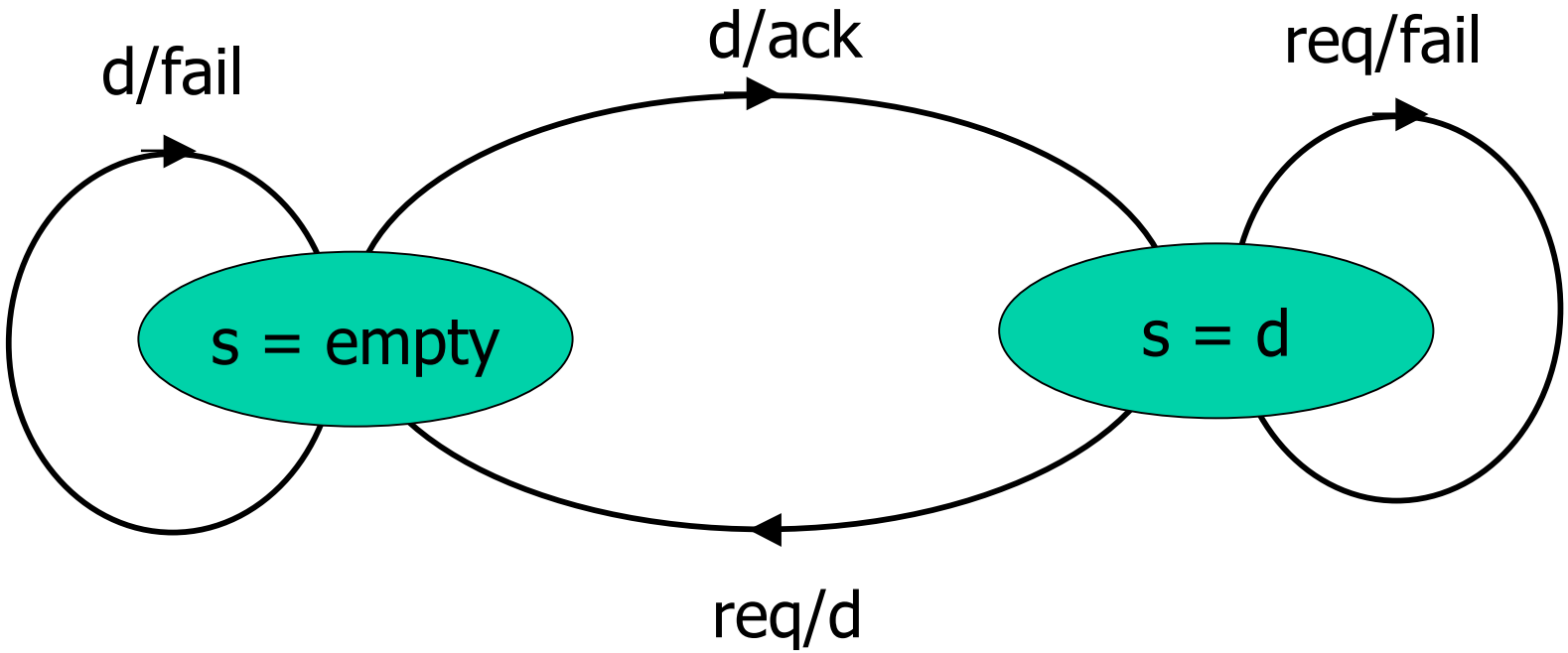
in x: QIn
out y: QOut

$$\{req\}\#x = (\{fail\}\#y) + (Data\#y)$$
$$\wedge\ \overline{y} \sqsubseteq Data \copyright \overline{x}$$
$$\wedge\ \forall\ t:\ \{req\}\#(x{\downarrow}t) \leq Data\#(x{\downarrow}t) + (\{fail\}\#(y{\downarrow}t))$$

The input assumption here is:

$$\forall\ t:\ \{req\}\#(x{\downarrow}t) \leq Data\#(x{\downarrow}t) + (\{fail\}\#(y{\downarrow}t))$$

A state machine for the Box



Assumption/Commitment Specifications

$F \in IF[I \blacktriangleright O]$

$A_t: \vec{I} \times \vec{O} \to \mathbb{B}$ input assumption at time t.

$A_t(x, y) = \exists\, x' \in IH[I], y' \in F.x' : x{\downarrow}t = x'{\downarrow}t \wedge y{\downarrow}t = y'{\downarrow}t$

$A_{t+1}(x, y) \Rightarrow A_t(x, y)$ for all $t \in \mathbb{N}$

$A: \vec{I} \to \mathbb{B}$ input assumption

$A(x) = \exists\, y \in IH[O] : y \in F.x \wedge \forall\, t \in \mathbb{N} : A_t(x, y)$

$A(x) \Rightarrow \exists\, y \in IH[O] : A_t(x, y)$ for all $t \in \mathbb{N}$

Assumption/Commitment Specifications

G_t: $\vec{I} \times \vec{O} \to \mathbb{B}$ output commitment at time t

$G_t(x, y) = \exists\, x' \in IH[I],\ y' \in F.x' : x{\downarrow}t = x'{\downarrow}t \wedge y{\downarrow}t+1 = y'{\downarrow}t+1$

$G_{t+1}(x, y) \Rightarrow G_t(x, y)$ for all $t \in \mathbb{N}$

$G_t(x, y) \Rightarrow A_t(x, y)$

G: $\vec{I} \times \vec{O} \to \mathbb{B}$ commitment

$G(x, y) = y \in F.x$

$G(x, y) \Rightarrow G_t(x, y)$ for all $t \in \mathbb{N}$

$G(x, y) \Rightarrow A(x)$

Assumption/Commitment Specification of Services

Service specification:

$$F.x = \{y: A(x) \wedge G(x, y)\}$$

$F.x = \varnothing$ iff $\neg A(x)$

Component specification:

$$F.x = \{y: A(x) \Rightarrow G(x, y)$$

$$\wedge\ \forall\, t \in IN: A_t(x, y) \Rightarrow G_t(x, y)\}$$

$F.x = (F.x)\downarrow t\hat{}\vec{O}$ "chaos" iff $\forall\, y: \neg A_t(x, y)$

Composition of Components and Services

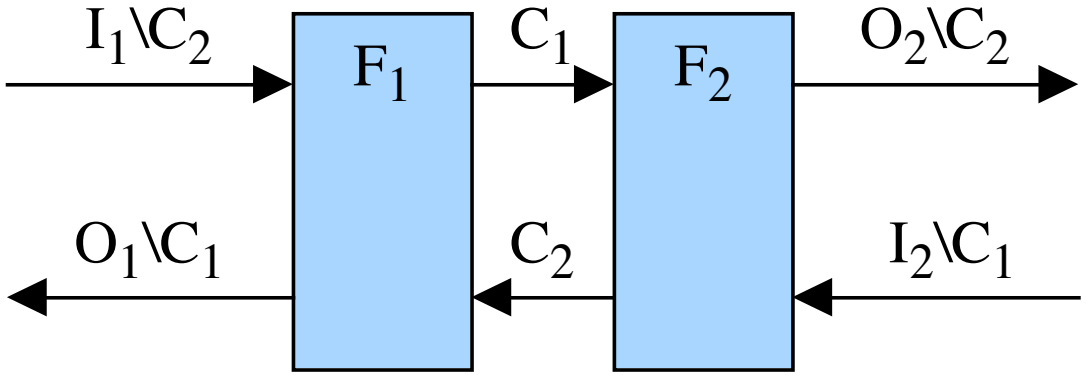
Given

$F_1 \in IF[I_1 \blacktriangleright O_1]$

$F_2 \in IF[I_2 \blacktriangleright O_2]$

$C_1 \subseteq O_1 \cap I_2$

$C_2 \subseteq O_2 \cap I_1$



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

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

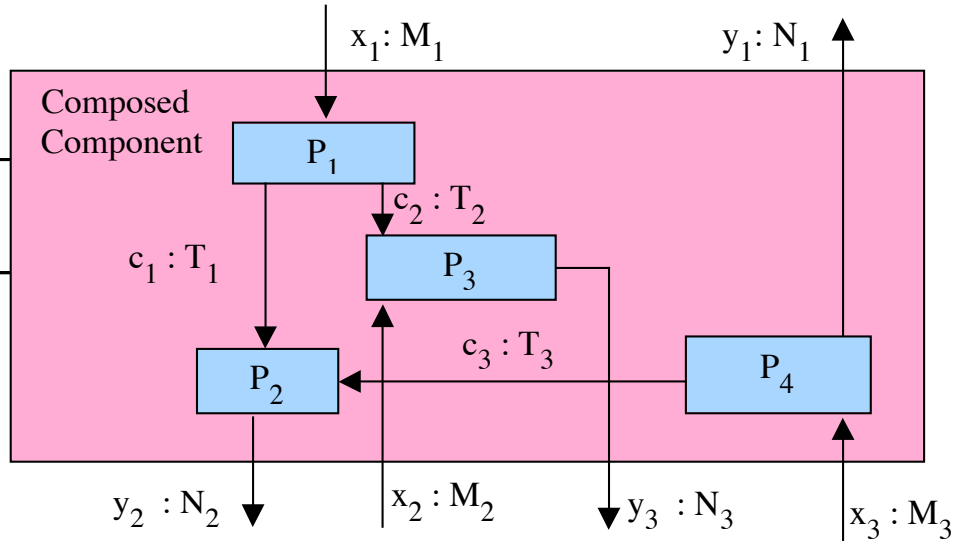
Architectures: Specification of composed components

Composed components spec

in $x_1: T_1 , x_2: T_2 , \ldots$

out $y_1: S_1 , y_2: S_2 , \ldots$

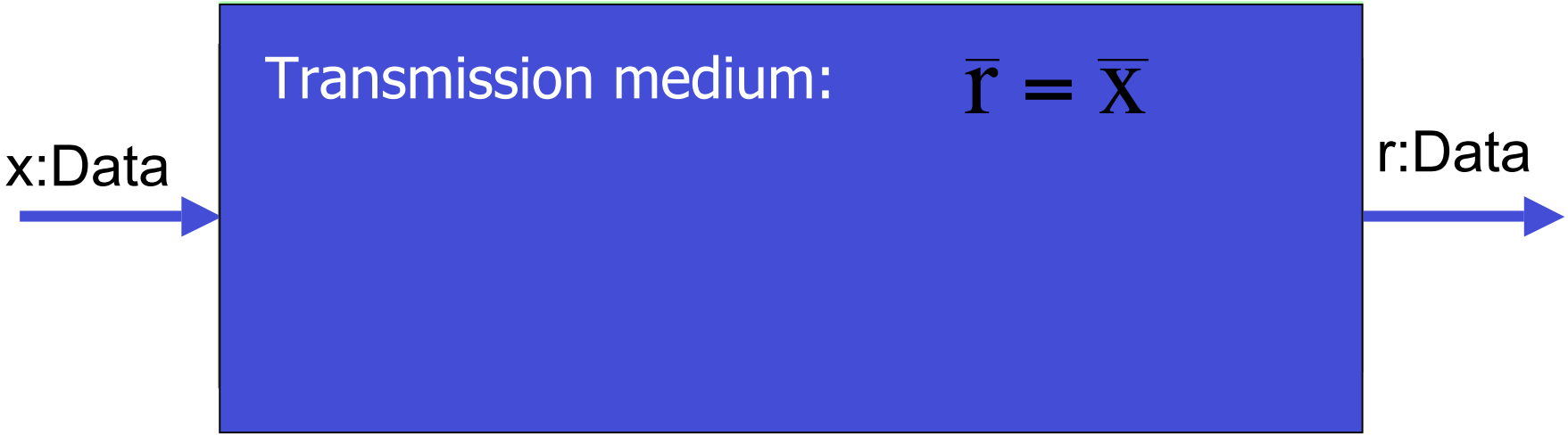
$\exists c_1, c_2 , \ldots : P_1 \wedge \ldots \wedge P_n$



Parallel composition = logical and

Channel hiding = existential quantification

Example: Alternating Bit Protokoll



Leads in the case of deterministic components to the equations :

c1 = sender(x, c4)

c2 = medium(c1)

(r, c3) = receiver(c2)

c4 = medium(c3)

In case of nondeterminism

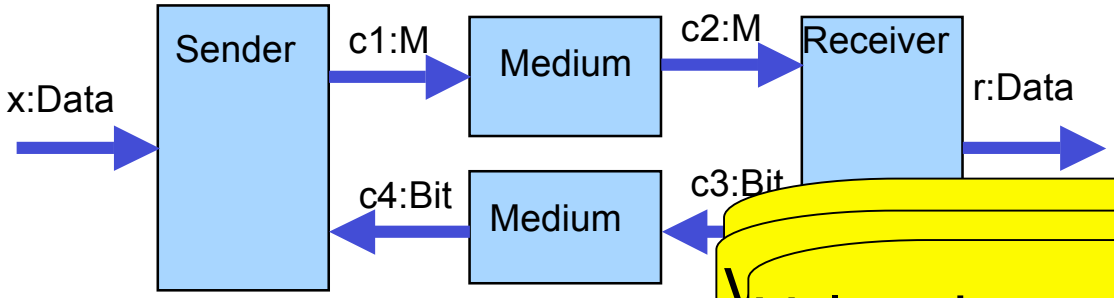
c1 ∈ sender(x, c4)

c2 ∈ medium(c1)

(r, c3) ∈ receiver(c2)

c4 ∈ medium(c3)

Fixpoints



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

Value does only depend on values in time intervals t < 2

Given stream x how

time	1	2	3	4	5	6	7	8	9	10	11
x	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩
c1	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩								
c2	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩								
c3	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩								
c4	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩								
r	⟨ … ⟩	⟨ … ⟩	⟨ … ⟩								

Property Refinement

Given two service interfaces $F1, F2 \in IF[I \blacktriangleright O]$

we call service $F2$ a property refinement of $F1$
if for all input histories $x \in IH[I]$

$$F2.x \subseteq F1.x$$

Then we write

$$F1 \equiv> F2$$

we get compositionality:

$$\frac{F1 \equiv> F2 \qquad G1 \equiv> G2}{F1[C1 \leftrightarrow C2]G1 \equiv> F2[C1 \leftrightarrow C2]G2}$$

Using Queues for Boxes

- We can use the component Queue wherever we need a component Box since:

Box ≡> Queue

since:

$$\text{req\#x} = \text{Data\#y} \wedge \overline{y} \sqsubseteq \text{Data}\copyright\overline{x} \Rightarrow$$
$$\text{req\#x} = \text{Data\#y} \wedge \forall\, d \in \text{Data}: \{d\}\#y \leq \{d\}\#x$$

To prove this implication proves the refinement relation.

Remarks

- Property refinement

  $F_1 \equiv\!> F_2$

  is easily proved by proving

  $P_2 \Rightarrow P_1$

  for the specifying assertion P_1 and P_2 of F_1 and F_2 resp.

- Property refinement is not appropriate for relating services with components (in property refinement the domain is always decreased)
  - ◇ One solution: chaos completion
  - ◇ Another solution: a more refined refinement relation