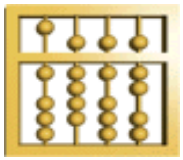
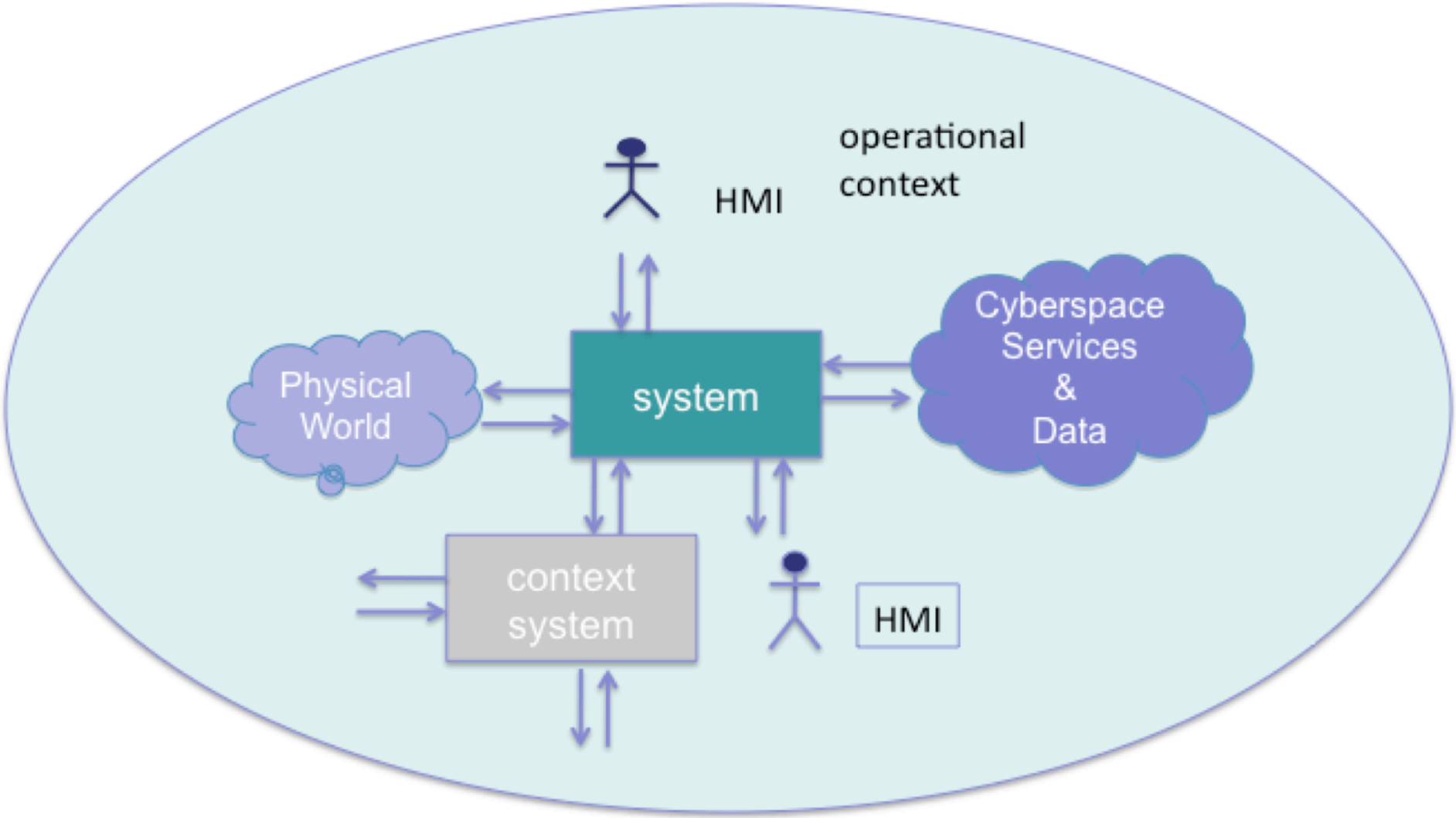
Contracts: Model-centric Assumption Promise System Specification

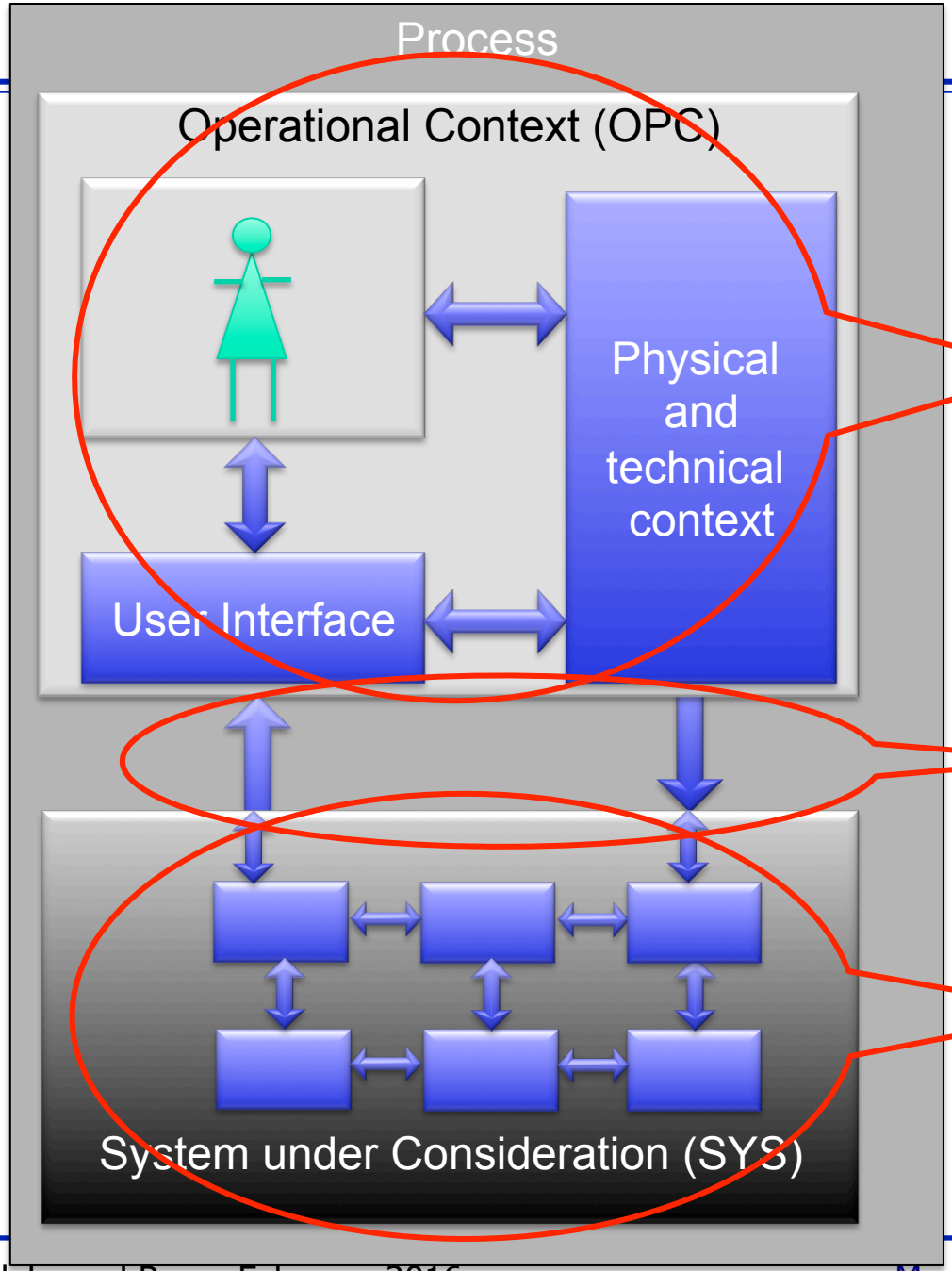
Manfred Broy

Technische Universität
München
Institut für Informatik



A System and its Operational Context





$$ASU \wedge PRO \Rightarrow IAS$$

Operational Context (ASU)
Assumptions for the requirements specification

Interaction Observations (IAS)
Properties for the observations in the requirements specification

System (PRO)
Properties promised in the system specification

Requirements Specification: Modeling SYS, OPC, OBS

- formulating system properties – system promises PRO
- formulating properties of the operational context – context assumptions ASU
- formulating properties of the interaction between the system and its operational context – interaction assertions IAS

$$ASU \wedge PRO \Rightarrow IAS$$

- This leads to assumption/promise specification formats

Discrete systems: the system modeling theory

Sets of typed channels

$$I = \{x_1 : T_1, x_2 : T_2, \ldots\}$$

$$O = \{y_1 : T'_1, y_2 : T'_2, \ldots\}$$

syntactic interface

$$(I \blacktriangleright O)$$

data stream of type T

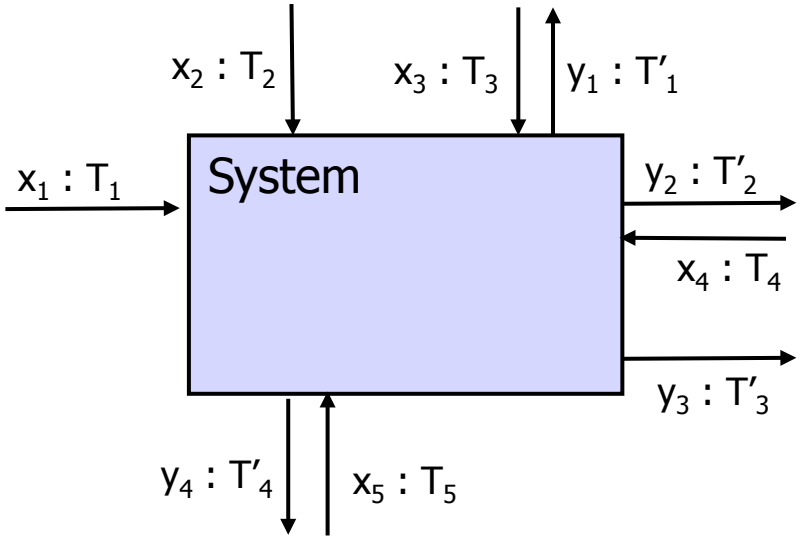
$$STREAM[T] = \{\mathbb{N}\backslash\{0\} \rightarrow T^*\}$$

valuation of channel set C

$$[C] = \{C \rightarrow STREAM[T]\}$$

interface behaviour for syn. interface $(I \blacktriangleright O)$

$$[I \blacktriangleright O] = \{[I] \rightarrow \wp([O])\}$$



Example: System interface specification

A question answering component

QAC

in x: Qst **out** y: Asw
∀ q ∈ Qst: q#x = A[q]#y

Every question gets answered!

Qst : set of questions
Asw : set of answers

A[q] : set of possible answers for question q

$q \neq q' \Rightarrow A[q] \cap A[q'] = \varnothing$

Verification: Proving properties about specified systems

From the interface assertions we can prove

- Safety properties

$$a \in y \wedge y \in QAC(x) \Rightarrow \exists\, q \in Qst:\ q \in x \wedge a \in A[q]$$

- Liveness properties

$$q \in x \wedge y \in QAC(x) \Rightarrow \exists\, a \in Asw:\ a \in y \wedge a \in A[q]$$

Example: Context interface specification

A context interface specification

CIS

in y: Asw **out** x: Qst
∀ t ∈ Time: Qst#(x↓t)+1 ≤ Asw#(y↓t)

Never ask a further question before your recent one is answered

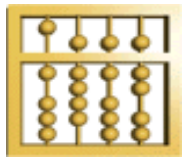
Example: QAC with Timing Restrictions

QAC

in x: Qst **out** y: Asw
$\forall\, t \in IN: \forall\, q \in Qst:$ $A[q]\#(y\downarrow t+delay)$ $\leq q\#(x\downarrow t)$ $\leq A[q]\#(y\downarrow t+delay+deadline)$

Universal Properties of System (Interfaces)

Technische Universität
München
Institut für Informatik



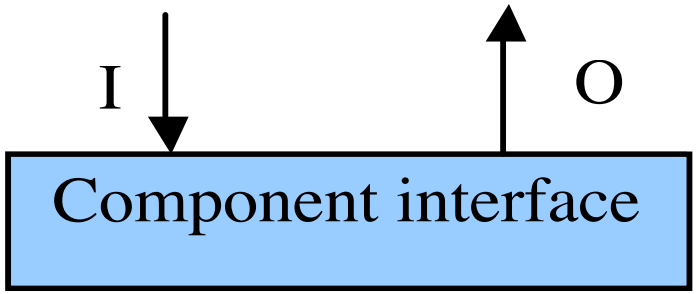
System interface behaviour - causality

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

$F \in [I \blacktriangleright O]$ *semantic interface* for $(I \blacktriangleright O)$ with *timing property addressing strong causality* (let $x, z \in [I]$, $y \in [O]$, $t \in \mathbb{N}$):

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

$x{\downarrow}t$ prefix of history x of length t



Verification: adding/exploiting causality

From the interface assertions we can derive properties!

Specification:

$$y \in QAC(x) \Rightarrow (\forall\, q \in Qst:\ \ q\#x = A[q]\#y\ )$$

Strong causality: $\forall\, t \in Time$:

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

From which by choosing z such that

$$\#(z{\uparrow}t) = 0$$

we can deduce (note then $q\#x{\downarrow}t = q\#z$)

$$y \in QAC(x) \Rightarrow \forall\, q \in Qst:\ A[q]\#(y{\downarrow}t+1) \leq q\#(x{\downarrow}t)$$

No answers before questions!

Causal deterministic behaviors

- A total function f: **[**I**]** → **[**O**]** is called *causal* (and *strongly causal*, respectively) if behaviour

$$F \in [I \blacktriangleright O] \text{ with } F(x) = \{f(x)\}$$

  is causal (or strongly causal, respectively) for all $x \in$ **[**I**]**
- A nondeterministic behaviour F defines the set **[**F**]** of total deterministic behaviours.

Realizability

An interface behaviour F is called (*strongly*) *realizable* if there exists a (strongly) causal "deterministic" function f: **[I]** → **[O]** such that

$$\forall\, x \in [\![I]\!] : f(x) \in F(x)$$

f is called (strong) realization of F.

Theorem

An interface behaviour F is (strongly) realizable if there exists a (Moore) Mealy machine that calculates F.

Example: Non-realizable Behaviour

Consider the behaviour $F \in [I \blacktriangleright O]$:

$$F(x) = \{y \in [O] : x \neq y\}$$

F is strongly causal but not realizable.

Proof: Strong causality is obvious.

If F were realizable $f \in [F]$ exists with

$$\forall\, x \in [I] : f(x) \in F(x)$$

Since f is strongly causal there exists a fixpoint z with $z = f(z)$.

By $f \in [F]$ we get by $y = f(x)$ the proposition $y \in F(x)$ and by the specification $x \neq y$ and thus for the fixpoint z the conclusion $z \in F(z)$ which yields $z \neq z$ and thus a contradiction.

Healthiness Conditions for System Specifications

Accordingly, for an interface assertion spc(x, y) the following healthiness conditions are required:

Systems does react not earlier as in the next time interval

Existential satisfiability: $\forall\, x: \exists\, y: spc(x, y)$

Strong causality in input x:

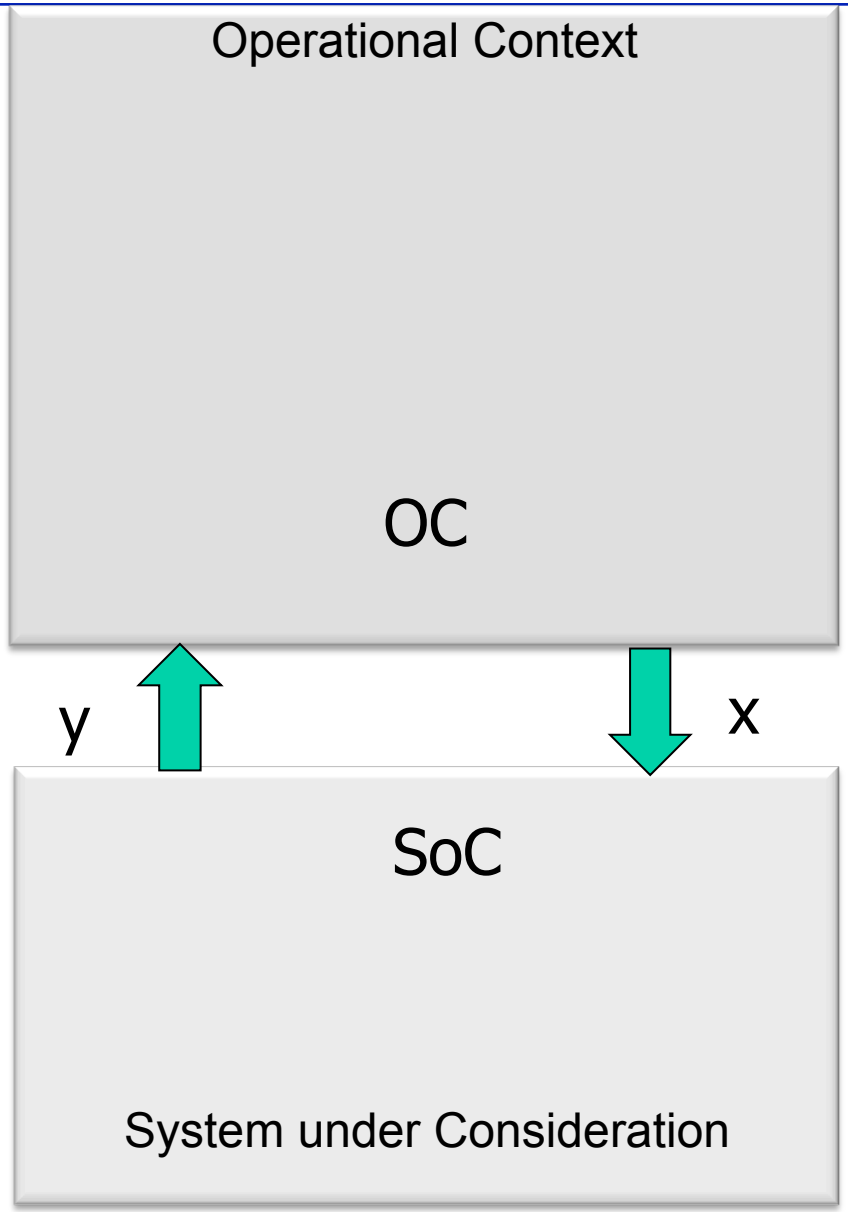
$$\forall\, x, x': \forall\, t: x{\downarrow}t = x'{\downarrow}t \Rightarrow \forall\, y: spc(x, y{\downarrow}t+1) = spc(x', y{\downarrow}t+1)$$

Realizability: $\exists\, f \in If_{sc}[I \blacktriangleright O]: \forall\, x: spc(x, f(x))$

Full realizability:

$$\forall\, x, y: spc(x, y) \Rightarrow \exists\, f \in If_{sc}[I \blacktriangleright O]: y = f(x) \wedge \forall\, x': spc(x', f(x'))$$

Modeling



We model

- context behavior by (weakly) causal behaviors assuming that contexts may react instantaneously – reaction within one time interval
- system by strong causal behavior – reaction requires at least one step in time

Consequence:
Unique fixpoints for realizations of the context and the system specs

Healthiness Conditions for System Context Specifications

Accordingly, for an interface assertion asu(x, y) of the context the following healthiness conditions are required:

Existential satisfiability: $\forall\, y: \exists\, x: asu(x, y)$

Context may react immediately in the current time interval

Weak causality in input x:

$$\forall\, y, y': \forall\, t: y{\downarrow}t = y'{\downarrow}t \Rightarrow \forall\, y: asu(x{\downarrow}t, y) = asu(x{\downarrow}t, y')$$

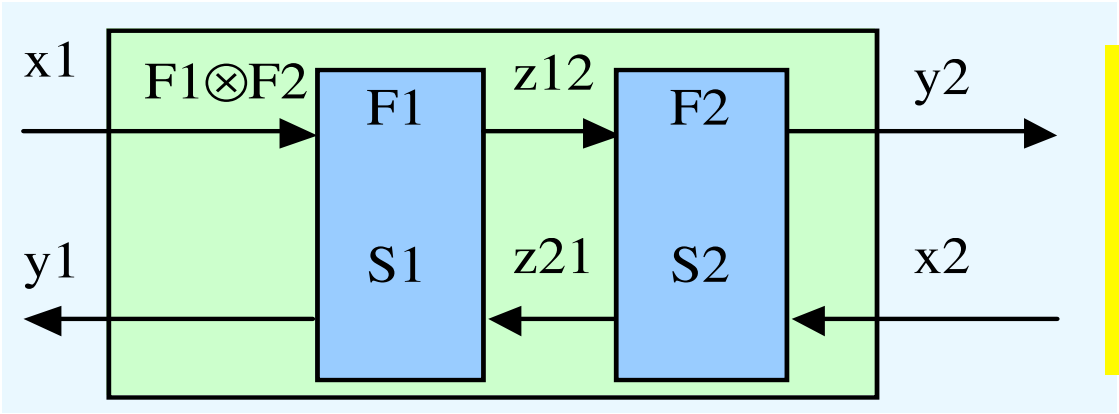
Realizability:

$$\exists\, g \in If_c[O \blacktriangleright I]: \forall\, y: asu(g(y), y)$$

Full realizability:

$$\forall\, x, y: asu(x, y) \Rightarrow \exists\, g \in If_c[O \blacktriangleright I]: x = g(y) \wedge \forall\, y': asu(g(y'), y')$$

Modularity: Rules of compositions for interface specs



We do not need assumptions to achieve modularity!

F1

in x1, z21: T **out** y1, z12: T
S1

F2

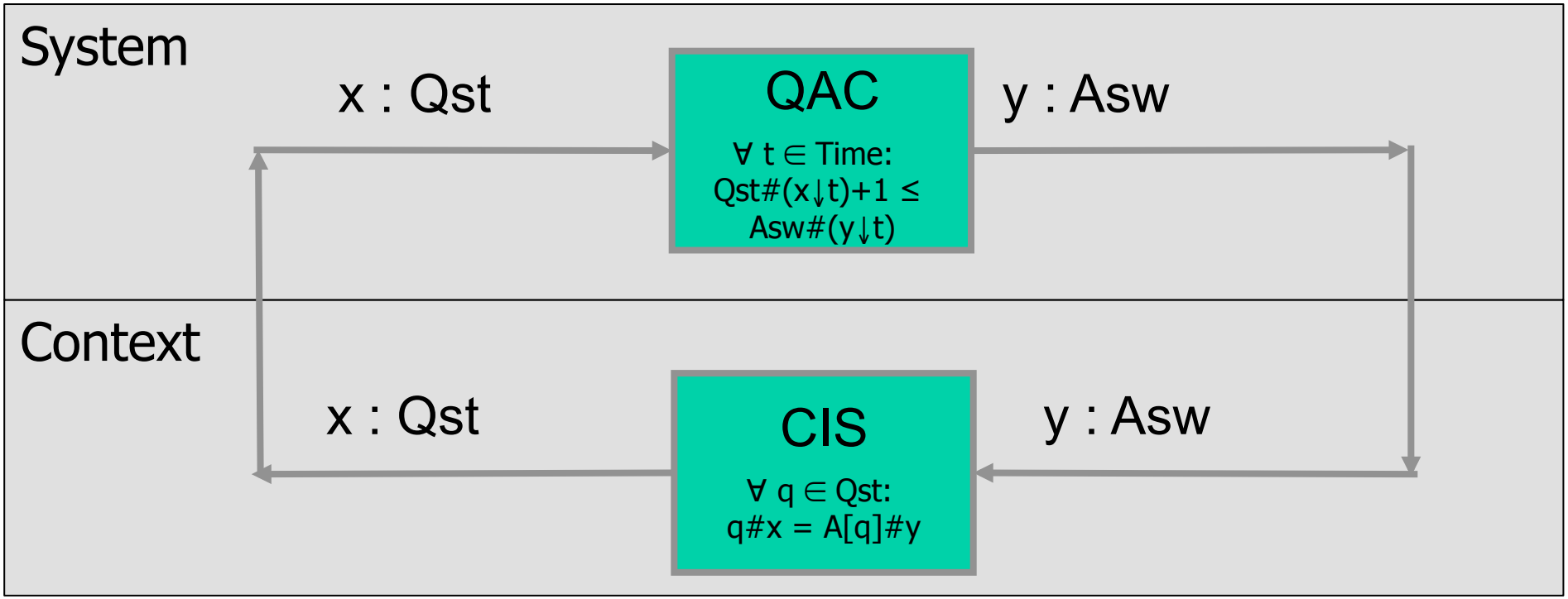
in x2, z12: T **out** y2, z21: T
S2

F1⊗F2

in x1, x2: T **out** y1, y2: T
∃ z12, z21: S1 ∧ S2

S1 ∧ S2 is called the interaction assertion

Qsts and answers

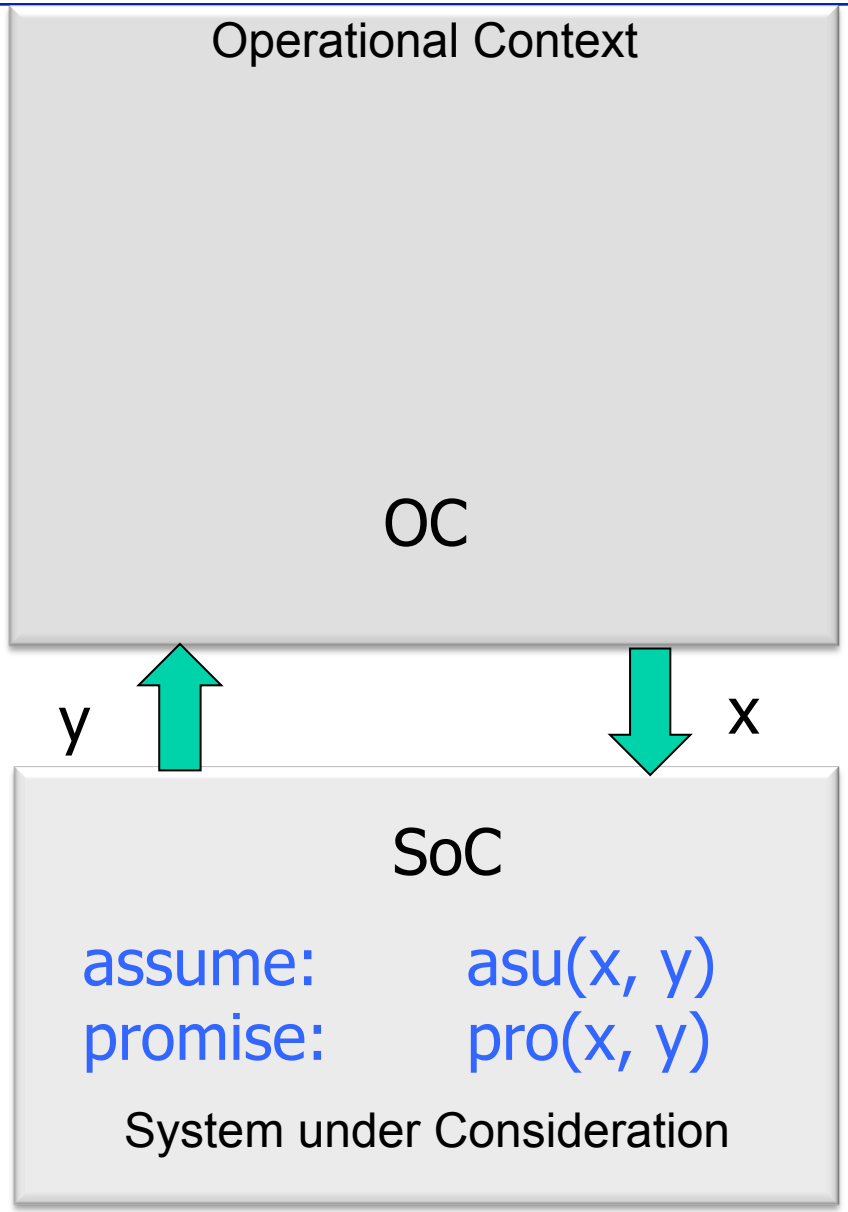


Interaction assertion:

∀ t ∈ Time: Qst#(x↓t)+1 ≤ Asw#(y↓t)

∧ ∀ q ∈ Qst: q#x = A[q]#y

Assumption/Promise (A/P) Specifications

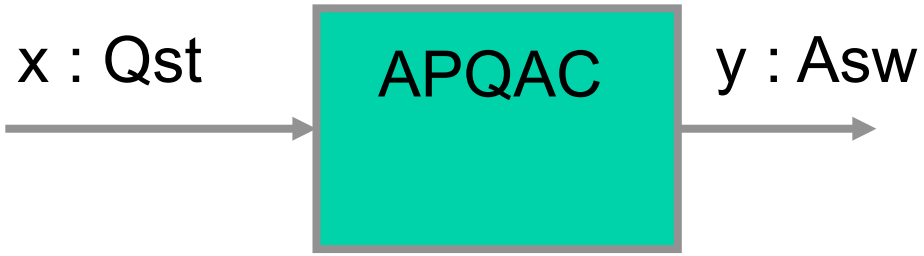


Assumption: asu(x, y)
The properties that we assume about the interface behavior of a context

Promise: pro(x, y)
The properties that are guaranteed about the interface behavior of the system

Resulting system spec:
asu(x, y) ⇒ pro(x, y)

Example: Assumption promise system interface specification



A contract for aquestion answering component

APQAC

in	x: Qst
out	y: Asw

assumption

$\forall\ t \in \text{Time}:\ \text{Qst}\#(x{\downarrow}t)+1 \leq \text{Asw}\#(y{\downarrow}t)$

promise

$\forall\ q \in \text{Qst}:\ q\#x = A[q]\#y$

Every question gets answered

-

as long as the next question is answered only after all questions have been answered!

What is a good
(a “healthy”) assumption?

Why Assumptions are Constraint by Output Histories

- In the general case, assumptions refer to output of the system.

  The reason is that if a system is nondeterministic and the question which input x fulfils the assumption may depend on the actual output y produced so far.

- Example: our QAS

$$asu(x, y) \equiv \forall\, t \in Time:\ \ Qst\#(x{\downarrow}t)+1 \leq Asw\#(y{\downarrow}t)$$

$$pro(x, y) \equiv \forall\, q \in Qst:\ q\#x = A[q]\#y$$

We obtain the specification in terms of an interface assertion

$$con(x, y) \equiv [asu(x, y) \Rightarrow pro(x, y)]$$

The assumption is fulfilled

- ◇ if a question is never sent as input to the system
- ◇ before the answer to the previously question has been returned by the system as output.

What makes an Interface Assertion a Healthy Assumption

- Assumptions should only constrain properties of the context.
  - ◇ In the case of simple assumptions that only refer to the input histories $x \in [I]$ for systems with systematic interface $(O \blacktriangleright I)$ this is obvious.
- However, what does it mean that asu only constraints the input histories for general assumptions.

$$asu : [I] \times [O] \to IB$$

Healthiness Conditions for Context Specifications

Accordingly, for an interface assertion asu(x, y) the following healthiness conditions are required:

Existential satisfiability: $\forall\, y: \exists\, x: asu(x, y)$

Causality in input y: $\forall\, y, y': \forall\, t : y{\downarrow}t = y'{\downarrow}t \Rightarrow \forall\, x: asu(x{\downarrow}t, y) = asu(x'{\downarrow}t, y)$

Realizability: $\exists\, g \in If_c[O \blacktriangleright I]: \forall\, x: asu(g(y), y)$

Full realizability: $\forall\, x, y: asu(x, y) \Rightarrow \exists\, g \in If_c[O \blacktriangleright I]: x = g(y) \wedge \forall\, y': asu(g(y'), y')$

Example: Implicative Assertions

Consider a system with one input channel x and one output channel y, both carrying natural numbers as messages.
Let n be a given natural number.

A specification in implicative form:

$$con(x, y) \equiv [n\#y = 0 \Rightarrow n\#x = 0]$$

Is n#y = 0 a healthy assumption about the context

Clearly, there does not exist a context that can guarantee the premise n#y = 0, since the output is exclusively determined by the system.

Example: Implicative Assertions

The A/P–specification

assume: $n\#y = 0$

promise: $n\#x = 0$

is not healthy, since

- the assumption does not constrain the input histories but the output.
- The promise $n\#y = 0$ is not healthy as an assumption, since it does not express properties of input stream x but only of output stream y.

The assertion $n\#y = 0$ is not causal in history y, since causality in y would require for all $t \in IN$

$$y{\downarrow}t = y'{\downarrow}t \Rightarrow \forall\, x: (n\#y = 0) \equiv (n\#y' = 0)$$

which does not hold.

Assertion $n\#y = 0$ is therefore not a *healthy* assumption, since it is not causal in y and thus not realizable by any context.

Example: Implicative Assertions

In the assertion (which is equivalent to assertion con(x, y) by contraposition)

$$con(x, y) \equiv [n\#x > 0 \Rightarrow n\#y > 0]$$

the assertion $n\#x > 0$ is causal in history y since the formula

$$y{\downarrow}t = y'{\downarrow}t \Rightarrow \forall\, x: (n\#x{\downarrow}t > 0) \equiv (n\#x{\downarrow}t > 0)$$

holds. It is furthermore trivially realizable.

This interface assertion may therefore be rewritten in the A/P-format of a contract

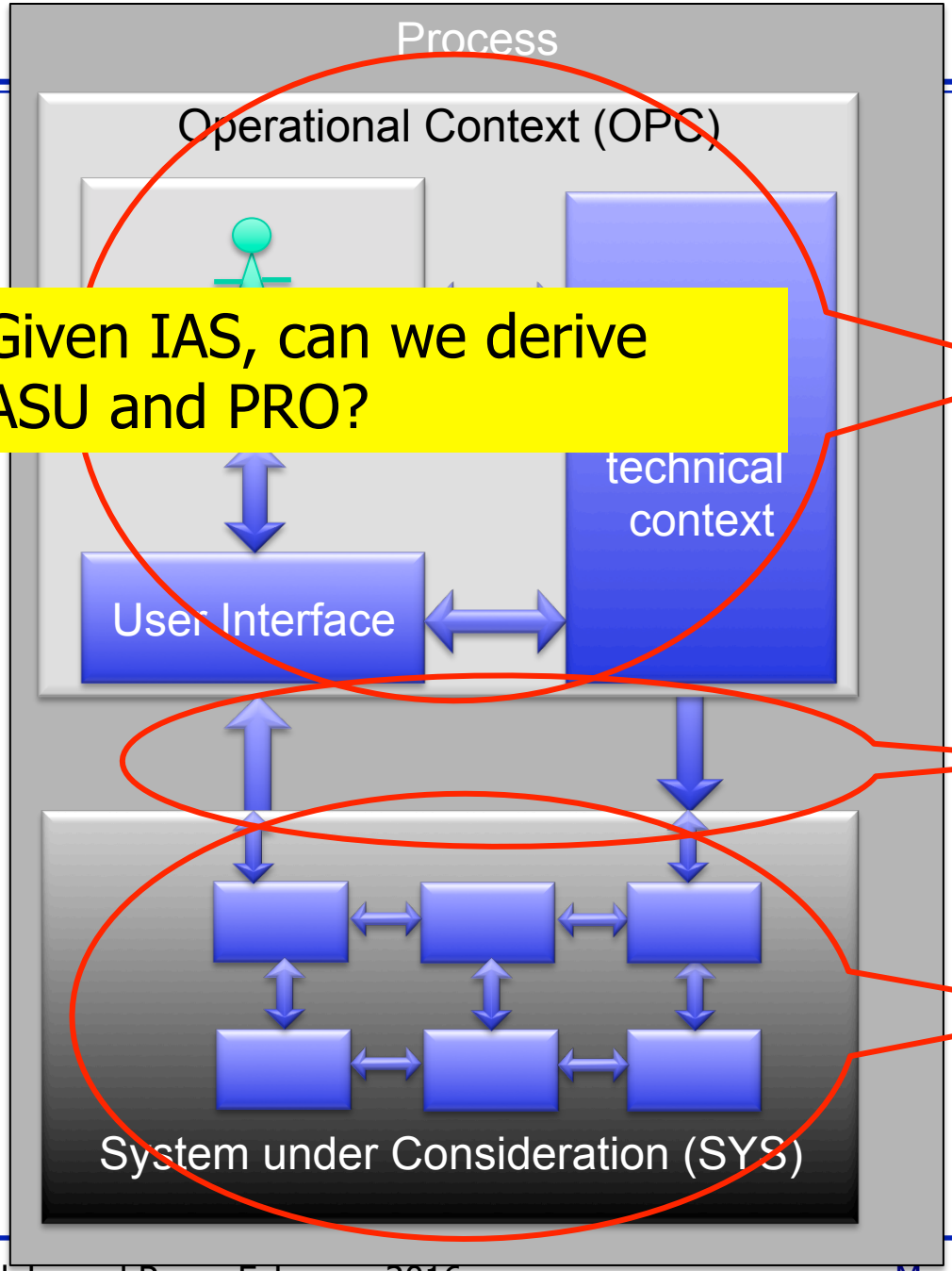
assume: $n\#x > 0$

promise: $n\#y > 0$

with a healthy assumption.

Conclusion: Not every assertion is a healthy assumption.

From Interaction Assertions to Assumptions and Promises



Given IAS, can we derive ASU and PRO?

ASU ∧ PRO ⇒ IAS

Operational Context (ASU)
Assumptions for the requirements specification

Interaction Assertion (IAS)
Properties for the observations in the requirements specification

System (PRO)
Properties captured in the system specification

Notation

Throughout the presentation we use the following notation:

Given a predicate

$$p: [C] \to IB$$

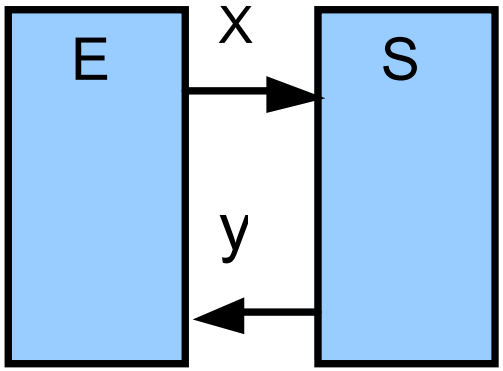
we extend for every time $t \in IN$ the predicate p also to finite histories x of length t:

$$p(x) \equiv \exists\, x' \in : x = x' \downarrow t \wedge p(x')$$

From Interaction Assertions to Contracts

Let the *interaction assertion*

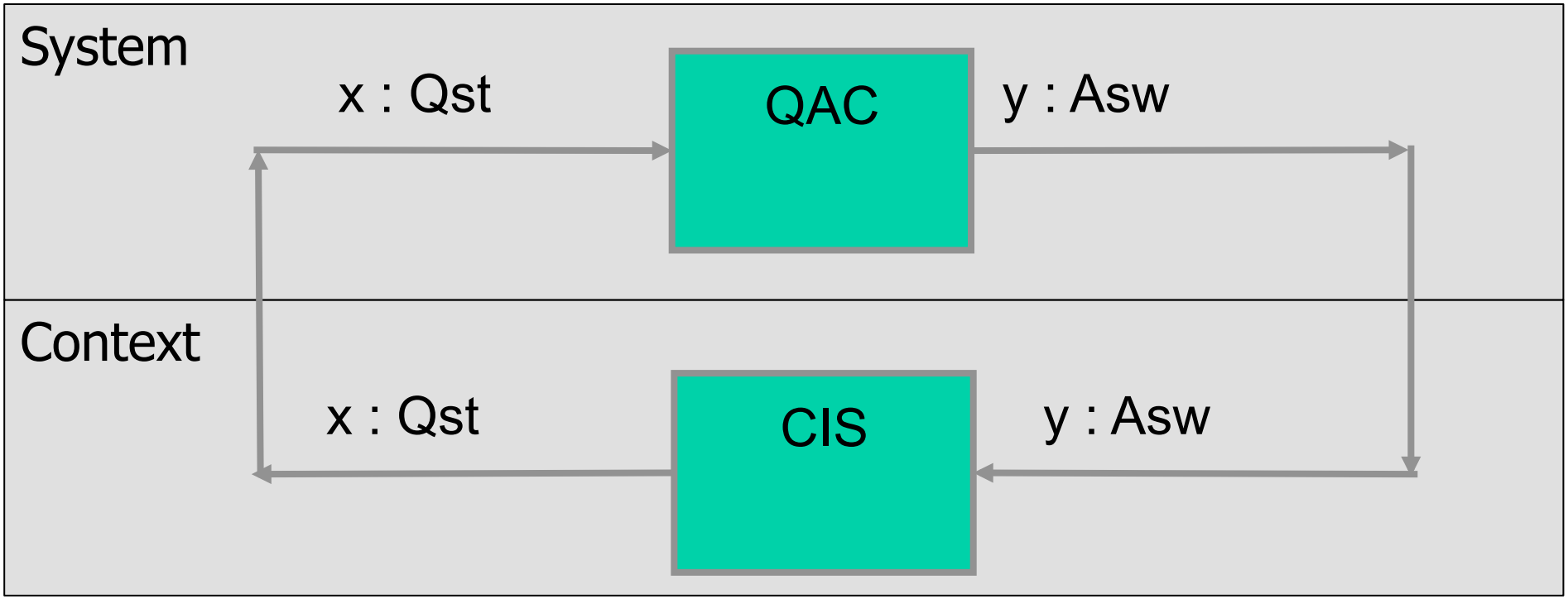
ias: **[I]** × **[O]** → IB



be given

ias(x, y) is an assertion characterizing the interaction between the system S and its context E in terms of the histories x and y.

Questions and answers



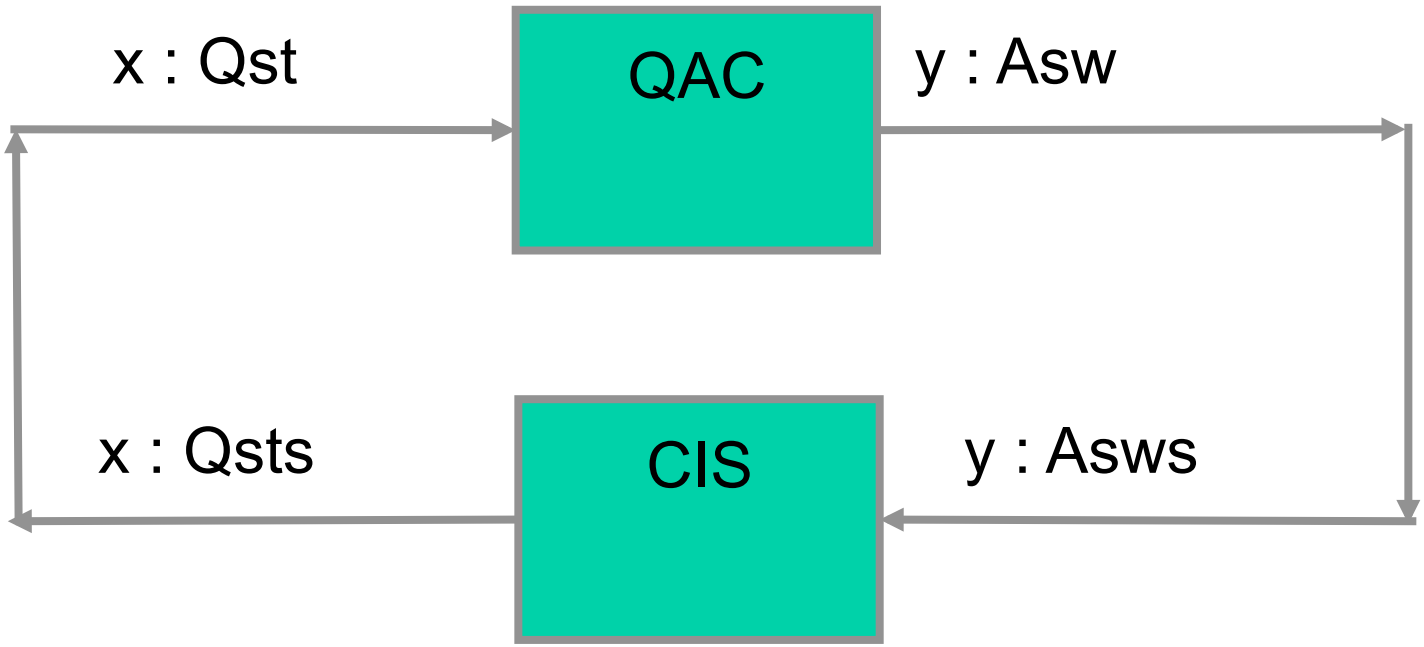
$$\forall\, t \in \mathrm{Time}:\ \mathrm{Qst}\#(x{\downarrow}t)+1 \le \mathrm{Asw}\#(y{\downarrow}t)$$
$$\wedge\ \forall\, q \in \mathrm{Qst}:\ q\#x = A[q]\#y$$

Deriving specs from interaction assertion

Can we derive from the interaction assertion:

$$\forall\, t \in \text{Time}:\ \ \text{Qst}\#(x{\downarrow}t)+1 \leq \text{Asw}\#(y{\downarrow}t)$$
$$\wedge\ \forall\, q \in \text{Qst}:\ q\#x = A[q]\#y$$

the contract in terms of assumptions and promises for



From Interaction Assertions to Contracts

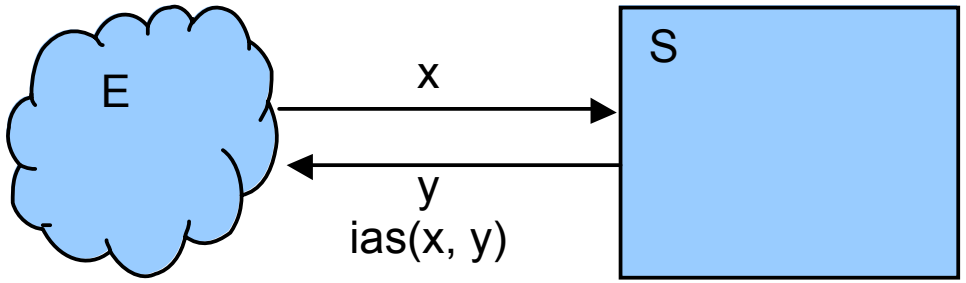
Given an interaction assertion ias(x, y) we derive an A/P-specification for system S with the weakest assumption by the following steps:

(1) Separate ias into a safety and a liveness part

(2) Separate the safety part of ias canonically into an assumption and a promise for system S

(3) Separate the liveness part of ias into an assumption and a promise for system S

(4) Construct a contact being the A/P-specification of S from the liveness and safety parts of the assumption and the promise.



From Interaction Assertions to Contracts: Safety

Deriving asu(x, y) and pro(x, y) from ias such that:

asu(x, y) ∧ pro(x, y): ias(x, y)

¬asu(x, y): ∃ t ∈ IN: ias(x↓t, y↓t+1) ∧ ¬ias(x↓t+1, y↓t+1)

¬pro(x, y): ∃ t ∈ IN: ias(x↓t, y↓t) ∧¬ias(x↓t, y↓t+1)

From Interaction Assertions to Contracts: Safety

Derive promise pro and a assumption asu from property ias

$$asu(x, y) \equiv [ias(x, y\downarrow 0) \wedge (\forall\, t: ias(x\downarrow t, y\downarrow t+1) \Rightarrow ias(x\downarrow t+1, y\downarrow t+1))]$$

$$pro(x, y) \equiv (\forall\, t: ias(x\downarrow t, y\downarrow t) \Rightarrow ias(x\downarrow t, y\downarrow t+1))$$

To eliminate partiality according to the input restriction in assertion ias(x, y) derive from interaction assertion ias(x, y) the weaker interface assertion con(x, y) specified by contract

$$con(x, y) \equiv [asu(x, y) \Rightarrow pro(x, y)]$$

An easy proof shows that con(x, y) is strongly causal.

From Interaction Assertions to Contracts: Safety

Note that according to our initial assumption interaction assertion ias(x, y) includes only safety properties.

Theorem:

With the definitions as given above we obtain under the condition that assertion ias(x, y) is a pure safety property

$$(asu(x, y) \wedge pro(x, y)) \Leftrightarrow ias(x, y)$$

From Interaction Assertions to Contracts: Liveness

If ias(x, y) includes nontrivial liveness conditions the separation into assumptions and promises of ias(x, y) is less canonical than for safety, in general.

- Some liveness conditions definitely formulate properties specifically about input histories x or histories y about output.
- There are liveness conditions that can not be canonically separated into assumptions and promises.
- **Example**: the assertion

$$\{1\}\#x + \{0\}\#y = \infty$$

can either be fulfilled by assuming an infinite number of copies of 1 in input history x or by promising an infinite number of copies of 0 in output history y.

From Interaction Assertions to Contracts: Liveness

Given an interaction assertion

$$ias(x, y)$$

that is a pure liveness condition we define an assumption asu_{ias} as follows

$$asu_{ias}(x) \equiv \exists\, y: ias(x, y)$$

and a promise pro_{ias} by the equation

$$pro_{ias}(x, y) \equiv ias(x, y)$$

Those parts of the liveness property ias that can either be fulfilled by the context or by the system under consideration are made part of the promise.

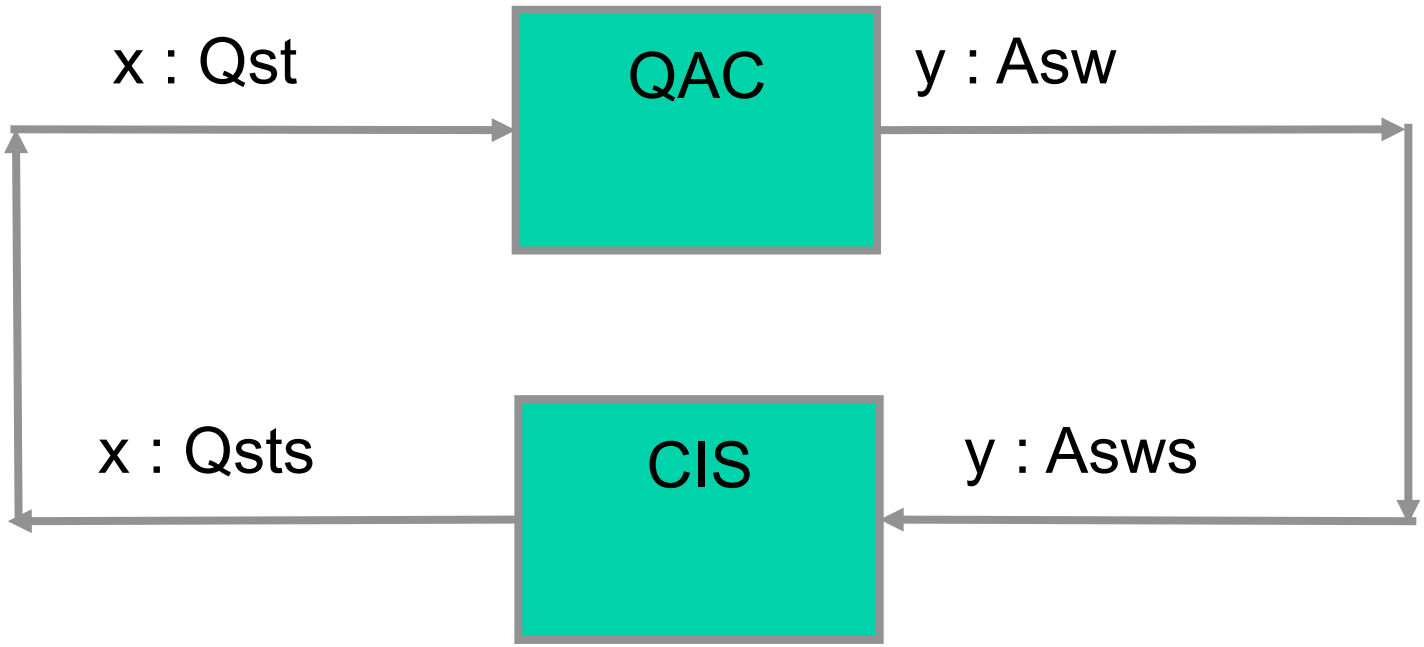
- This way we get the weakest assumption and the strongest promise for liveness properties of ias.

Deriving specs from interaction assertion

We derive from the interaction assertion:

$$\forall\, t \in \text{Time}:\ \text{Qst}\#(x{\downarrow}t)+1 \leq \text{Asw}\#(y{\downarrow}t)$$
$$\wedge\ \forall\, q \in \text{Qst}:\ q\#x = A[q]\#y$$

the specs for



Deriving specs from interaction assertion

We derive from the interaction assertion:

$$\forall\, t \in \text{Time}:\ \text{Qst}\#(x{\downarrow}t)+1 \le \text{Asw}\#(y{\downarrow}t)$$
$$\wedge\ \forall\, q \in \text{Qst}:\ q\#x = A[q]\#y$$

the specs for the system interface and the context:

Safety property in x and y

$$\forall\, t \in \text{Time}:\ \text{Qst}\#(x{\downarrow}t)+1 \le \text{Asw}\#(y{\downarrow}t)$$

is clearly an assumption.

The property

$$\forall\, q \in \text{Qst}:\ q\#x = A[q]\#y$$

is composed of a system safety property (by causality)

$$\forall\, t \in \text{Time}:\ \forall\, q \in \text{Qst}:\ q\#x{\downarrow}t \ge A[q]\#y{\downarrow}t+1$$

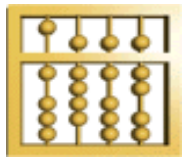
that is clearly a promise and liveness property

$$\forall\, q \in \text{Qst}:\ q\#x \le A[q]\#y$$

that is turned into a promise.

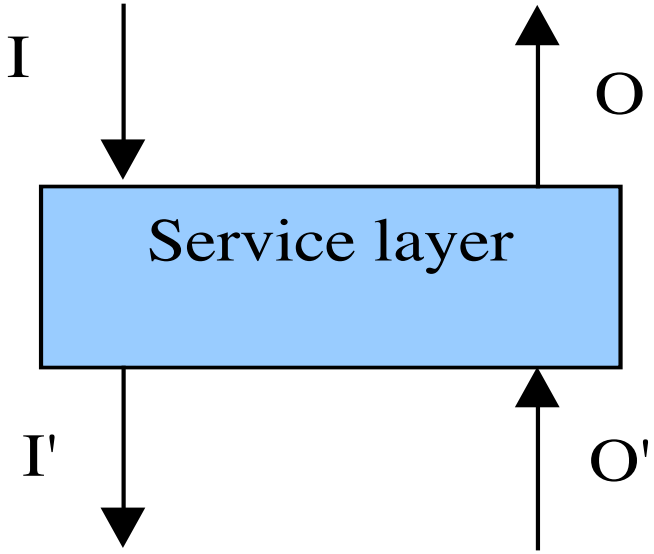
Service Layers and Service Stacks
Assumed and Promised Services

Technische Universität
München
Institut für Informatik



Service Layers

A service layer is a service
with syntactic interface (I∪O′ ▶ I′∪O)
structured into an
promised ("exported") service (I▶O)
assumed ("imported") service (I′▶O′).



We assume I ∩ O' = ∅ and O ∩ I' = ∅.

A service layer is a service with the interface behavior

L ∈ **[**I∪O′ ▶ I′∪O**]**

where both input and output actions are disjoint sets.

Service Layers

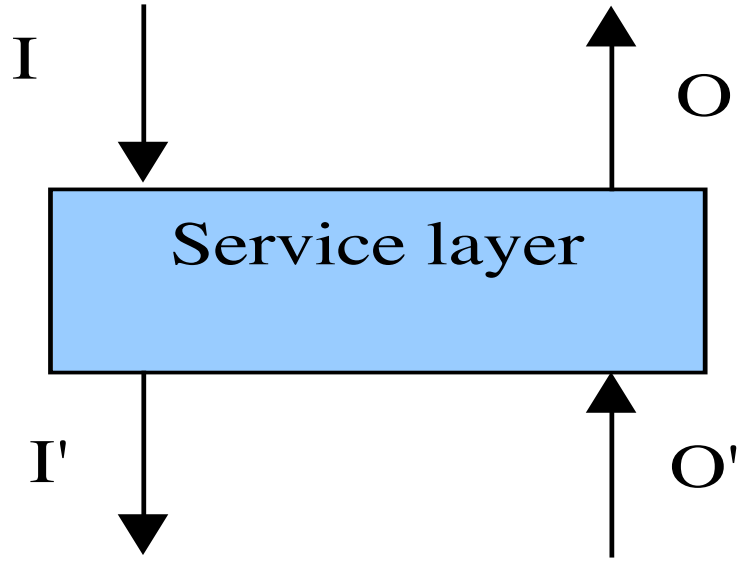
We denote the syntactic interface

of a service layer by

$(I \blacktriangleright O / O' \blacktriangleright I')$ syntactic service layer interface

The service layer is a service

$L \in [I \cup O' \blacktriangleright O \cup I']$

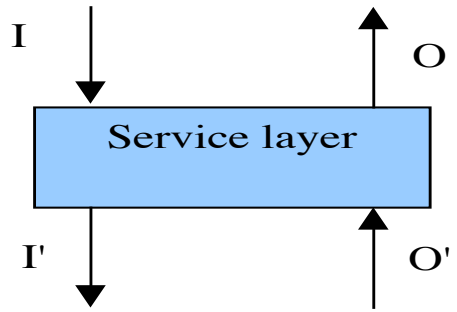


Specifying Service Layers

To specify a service layer we specify two services

The promised service $F \in [I \blacktriangleright O]$

The assumed service $R \in [I' \blacktriangleright O']$



A layer provides the service F under the condition that it gets the service R from "below".

Note that

- R does not specify the service of the layer as promised by L but the assumed service.

Given F and R we denote the layer L that offers service F provided service R is offered as an auxiliary service by

F//R

Specifying Service Layers

Layer

$$L = F//R$$

is specified as follows (for $x \in [I \cup O']$

$$L(x) = \{y \in [I' \cup O] : x|O' \in R(y|I') \Rightarrow y|O \in F(x|I)\ \}$$

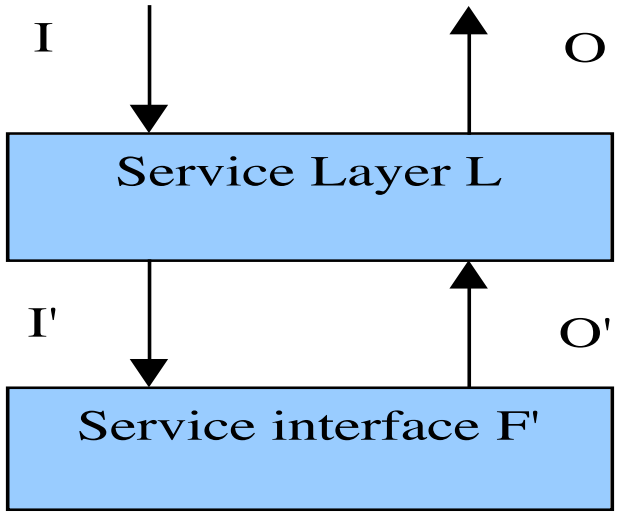
This expresses that

- if the service assumed from "below" is correct as required and specified by R than the offered service is as promised by F.
- Note that this specification is written in the pattern of an assumption/promise specification.

Composing Layers with Services

Given a service R' as requested from below

$R' \in [I' \triangleright O']$ imported service



and a service layer

$L \in [I \triangleright O/O' \triangleright I']$

where $L = F//R$

with given $F \in [I \triangleright O]$ and $R \in [I' \triangleright O']$

We get the composition of layer L with service R'

$L \otimes R' \in [I \triangleright O]$ composition of layer L with service R'

Service Refinement

Given services

$$F, F' \in [I \triangleright O]$$

F' is called a refinement of F iff

$$\forall\, x \in [I]: F'(x) \subseteq F(x)$$

then we write

$$F \rightarrowtail F'$$

Composing Layers with Services

$L \otimes R' \in [I \triangleright O]$ is a refinement of the provided service $F \in [I \triangleright O]$

$$F \rightarrowtail L \otimes R'$$

provided service R' is a refinement of the requested service R

$$R \rightarrowtail R'$$

Given layer $L = F//R$ we get the following proof rule for layered architectures

$$L = F//R \wedge R \rightarrowtail R' \Rightarrow F \rightarrowtail L \otimes R'$$

Composing Layers

Given two layers

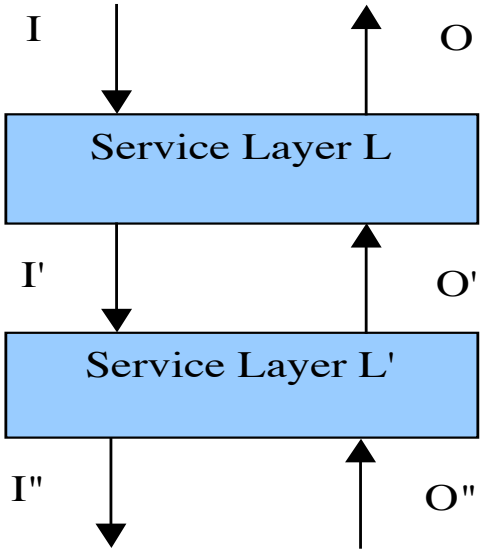
$$L \in [I \triangleright O / O' \triangleright I'],\ L' \in [I' \triangleright O' / O'' \triangleright I'']$$

where we assume that $I \cap I'' = \varnothing$ and $O \cap O'' = \varnothing$;

we define the layer composition

$$L \otimes L' \in [I \triangleright O / O'' \triangleright I'']$$

yielding a layer in $[I \triangleright O / O'' \triangleright I'']$.



Assume service L and L′ are described as follows

$$L = F // R$$

$$L' = F' // R'$$

We call these two layers fitting if

$$R \rightarrowtail F'$$

then we conclude $F // R' \rightarrowtail L \otimes L'$

proof rule for layered architectures

$$L = F // R \wedge L' = F' // R' \wedge R \rightarrowtail F'$$

$$\Rightarrow$$

$$F // R' \rightarrowtail L \otimes L'$$

Assumption/Promise Contracts at several levels

- Assumptions about input
  - ◇ What are healthy assumptions?
- Assumptions about the behavior of the operational context
  - ◇ What are generic properties of the operational context (what is the model of context behavior)?
- Decomposing interactions into assumptions and system properties
- Assumptions in architectures
  - ◇ Show that validity of assumptions of the system guarantee all assumptions of components of the system the system!
- Assumptions about required services
  - ◇ Specifying and composing service layers!