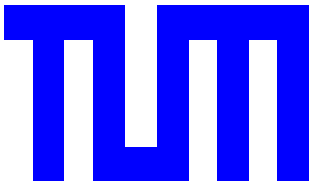
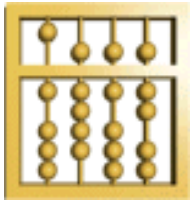
Time and Causality in Interactive Distributed Systems

Manfred Broy



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

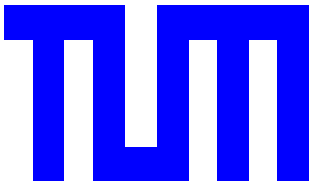


The Lectures I- IV

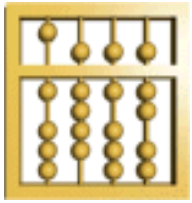
- Causality and Time in Discrete Systems
  - Discrete system models
  - Causality in systems
  - The role of time in system modeling
- A Modular System Model including Time and Causality
  - System interface behaviors
  - Specification
  - Composition
  - Refinement
- Changing the Granularity of Time
  - A flexible model of time
  - Time abstraction
  - Rules for Composition and Refinement
  - Causal Fixpoints and Causal Loops
- Delay Calculus
  - Flexible Timing of Systems
  - Composition and Delay
  - Optimal Delay Profile

Lecture I
Causality and Time in Discrete Systems

Manfred Broy



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



What is a System?

- System (from Latin systēma, in turn from Greek σύστημα systēma) is a set of interacting or interdependent entities, real or abstract, forming an integrated whole.
  - ◇ This means that a system has a border
  - ◇ That interactions take place - inside the system or between the system and its environment
  - ◇ A system's behaviour can be represented by a set of processes

What is a System?

- In physics the word system has a technical meaning, namely, it is the portion of the physical universe chosen for analysis.
  - ◇ Everything outside the system is known as the environment, which in analysis is ignored except for its effects on the system.
  - ◇ The cut between system and environment is a free choice, generally made to simplify the analysis as much as possible.
  - ◇ An isolated system is one which has negligible interaction with its environment - a closed in contrast to an open system.

Discrete Event Systems

- In physics, system behaviors are often modeled by continuous functions depending on time.
  - ◊ There, the dynamics of a system is captured by a set of variables that change their values continuously over time.
  - ◊ Dependencies between these variables are captured by continuous functions and expressed by formulas of differential calculus and integration theory, where time is represented by real numbers.
  - ◊ The values of the continuous functions represent the states of the system at the corresponding points in time.
- Discrete event systems provide a more abstract logical model of technical or economical systems:
  - ◊ Discrete steps of the systems modeled by events capture the dynamics of digital systems and discrete state changes.

Example: Adaptive Cruise Control (ACC)

- Continuous model:

  The negative/positive acceleration a of the car is a continuous function of the distance d to the successor car and the current speed s:

  $$a(t) = f(d(t), s(t))$$

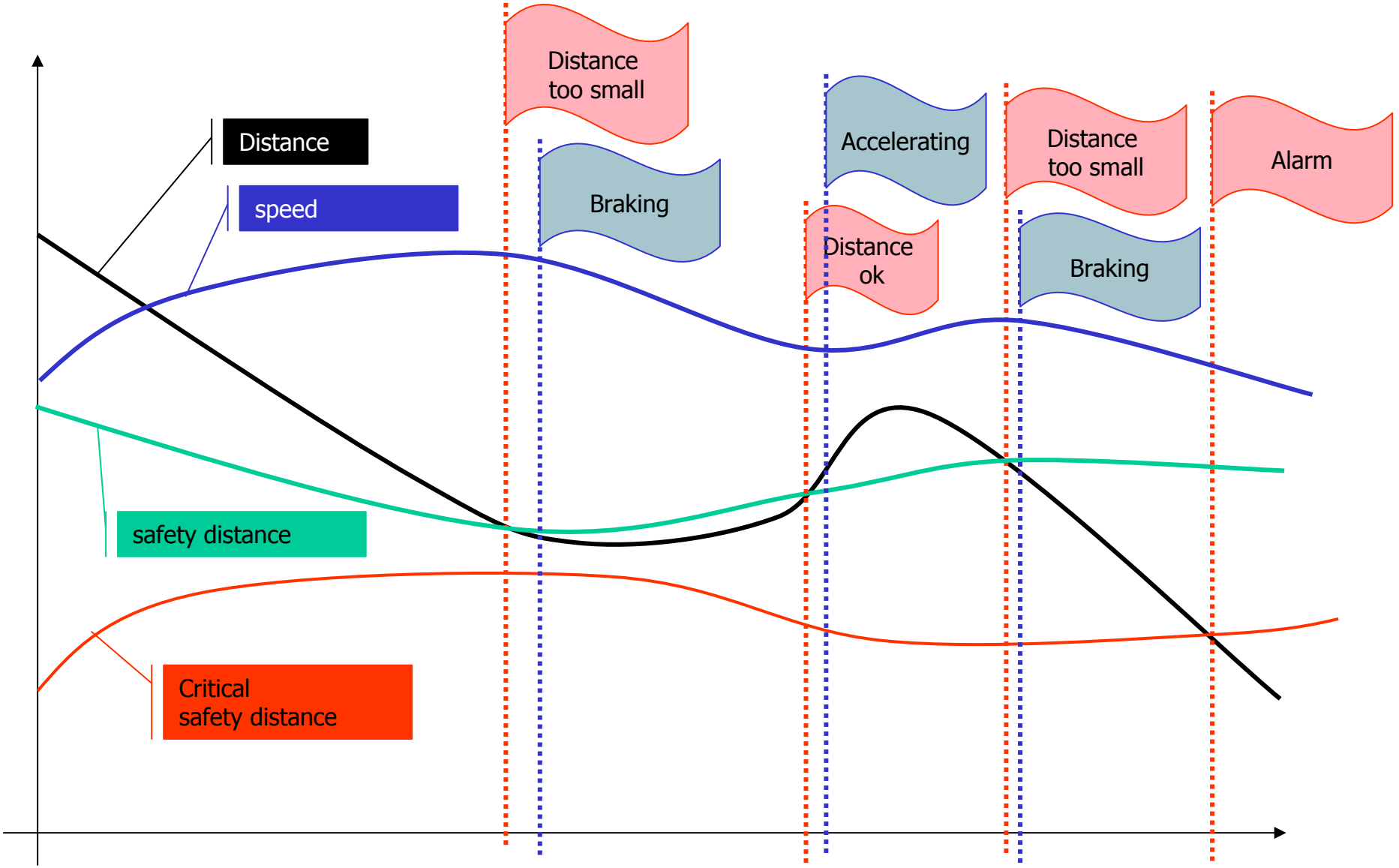
- Discrete model:

  If the distance d to the successor (relative to the current speed s) is below the target value, the brake is activated:

  $$d(t) < w(s(t)) \Rightarrow \text{braking}$$

  $$\text{distance too small} \Rightarrow \text{braking}$$

Continuous and Discrete Modeling



What is a process

- A process is a run (an execution) of a system
- For discrete systems the runs are discrete processes
- A system is in turn modelled by a set of processes
- A digital process is finite or even an infinite set E of discrete events.
- With each event some (or) action is related

Tony Hoare:

a trace

System Behaviors: Processes

- A discrete system shows a set of behaviors represented by processes.
  - ◊ In the literature a discrete system is often described by a state transition machine, for instance, or by a set of concurrent cooperating state machines, or by a Petri-Net.
  - ◊ When executing such a state machine, actions are carried out that correspond to state transitions.
  - ◊ Each instance of an action is an event.
    - Thus we associate for an *interpreted* process P (generated by the system) an action with each of its events.

System Behaviors: Processes

- Let E_p be the set of events forming a process P. The concept of a discrete interpreted process is defined by a mapping

  $$act: E_p \rightarrow Action$$

  where Action is the set of actions of the system.

  - ◇ Each action defines a state change, a timing action or a communication action.
  - ◇ This leads to a slightly more sophisticated notion of a system behavior represented by a set of processes where each event is timed and corresponds to an action.

Time

- Timing properties of systems can be classified into quantitative and qualitative aspects.
- The quantitative aspects aim at measuring time and talk about time distances between discrete events leading to questions such as:
  - ◇ How much time does it take until event A happens?
  - ◇ How much time does it take until event B happens after event A had happened?
- If we are not interested in measures of time distances between events as in hard real time applications there remain questions addressing the qualitative nature of timing properties, expressed by the after/before relation.
- Given such relations we can ask questions such as:
  - ◇ Does event A happen after event B?
  - ◇ Do events A and B happen simultaneously?
  - ◇ Does event A always happen only after event B?

Selecting Models of Time

- As a first critical question for system models we address the nature of time and its models:
  - For which engineering tasks and when is it necessary or at least more convenient to work with continuous, dense time (think about the reals IR) and when is a model of discrete time (think about the naturals IN) good enough or even more appropriate?
- We work with discrete time in the following!
  - Discrete time is typical for digital systems that are pulse driven and proceed in discrete time steps.
  - We want to demonstrate in the following that also discrete time allows for time models that are as flexible as continuous time.

Timed processes

- For each *timed* process formed by a set of events E their timing is a mapping

  time: $E \rightarrow TIME$

- If for the time domain TIME a linear order $\leq$ is given, a partial order $\angle$ (dependency order) is induced by the mapping time onto the sets of events ($e1, e2 \in E$):

  $e1 \angle e2 \Leftrightarrow (time(e1) < time(e2)) \vee e1 = e2$

- We may consider processes with a partial order $\angle$ on the set of events without specifying time

Question

- Are there dependency orders that are not induced by time?

Discrete Timed Processes

- Let E be a set of timed events; then each subset

  $P \subseteq E$

  defines a discrete timed (sub-)process.
- If the set of events is finite then P is called a finite process; otherwise it is called infinite.
- By TPROCESS we denote the set of all timed processes.

Observations about a system

- We assume that at each time t we observe a finite family of events.
  - ◊ In each run of the system we make a sequence of observations, one at each time point.
  - ◊ The timing introduces a structure on the events of a process. For each time $t \in TIME$ we define the sub-process (the partial process)

    $$\{e \in P: time(e) \leq t\}$$

    that consists of all events that take place till time t. This process is denoted by $P{\downarrow}t$.

Observations about a system

- A logical property of a timed process is represented by a predicate

  $$Q: \text{TPROCESS} \to \mathbb{IB}$$

  A property Q that holds at time t is a predicate on (finite) processes applied to the set P↓t of events.

  - ◇ Each predicate Q of that type applied to a process P↓t is called an observation about a system or about a process till time t.
  - ◇ We write also $Q_t(P)$ for Q(P↓t).

Temporal Operators

- We get a kind of temporal logic if we define (for a given process P)

  $\Diamond\ Q \equiv \exists\ t \in \text{TIME}: Q_t(P)$

  $\Box\ Q \equiv \forall\ t \in \text{TIME}: Q_t(P)$

  By these formulas we can introduce the temporal operators in a rather straightforward manner.

Stable observation

- An observation (a predicate) Q is called *stable*, if the following proposition

  $$Q_t(P) \wedge t < t' \Rightarrow Q_{t'}(P)$$

  holds for all processes P and all times t, t'.

Causality

- Causality is a notion that is more sophisticated than that of time.
  - Time is used to capture straightforward observations about systems.
  - Referring to causality we speak about the rules ("the logics") for the occurrence of events and actions in the executions (processes) of systems.
  - Timing is an observation about a single run representing a run of a system.
  - Causality speaks about the properties of all processes of a system and therefore deals with the system.
  - Causality addresses the rules and

Tony Hoare:

Dependence

Causality

- Causality addresses the logical dependencies between the events and actions in the digital processes of a concurrent interactive system S.
- Causality actually deals both with liveness and safety in observations:
  - ◇ *Liveness*: Which observation A about the system S guarantees another observation B to follow later eventually: we say "in system S observation A **leads_to** observation B".
  - ◇ *Safety*: Which observation B for the system S occurs only if another observation A was observed before: we may write "in system S observation B $\Rightarrow$ observation A"; we write B **requires** A
  - ◇ however, this expression using implication does not properly express the relation between the two observations stating that B follows only after A, since the causal relationship is expressed rather implicitly this way.

Example. Airbag

- An airbag in a car is activated only if the crash sensor indicated a crash and whenever the crash sensor indicates a crash it is activated.
- Both observations are stable.
  - ◇ Therefore if we do not consider time flow the two propositions: "crash sensor indicates crash" (csic) and "airbag is activated" (aia) are logical equivalent for completed processes.
- For time flow we require that

$$\forall\ t:\ csic(P{\downarrow}t) \Leftarrow aia(P{\downarrow}t)$$

$$\forall\ t:\ csic(P{\downarrow}t) \Rightarrow \exists\ t':\ t \leq t' \wedge aia(P{\downarrow}t')$$

Example. Air Bag (continued)

- For instance, in the case of an airbag

  (*) crash_sensor_indicates_crash $\Rightarrow$ airbag_is_activated

- This system property expresses that for each (complete) process of the airbag system this implication is valid.
  - ◇ if crash_sensor_indicates_crash, this leads_to airbag_is_activated.
- For the airbag we also assume the validity of the formula:

  airbag_is_activated $\Rightarrow$ crash_sensor_indicates_crash

- Now the "leads-to" interpretation is no longer appropriate. A more accurate interpretation is: "If the air bag is activated then the crash sensor has indicated a crash before."

Causality contrasted with conditionals

- Conditional statements are not statements of causality.
  - Perhaps the most important distinction is that statements of causality require the antecedent to precede the consequent in time, whereas this temporal order is not required by a conditional statement.
  - Since many different statements may be presented using "If...then..." in English (and, arguably, because this form is far more commonly used to make a statement of causality), they are commonly confused; they are distinct, however.

Stable Observations, Causality, and Conditional

- An observation is called *stable* for a process P, if in terms of temporal logic

  $\Box (A \Rightarrow \Diamond A)$

- If A is causal for B and if A and B are stable then for each complete process P system causality boils down to implication:

  $A(P) \Rightarrow B(P)$ and $B(P) \Rightarrow A(P)$

- Here logically there is no asymmetry between A and B, which indicates that by simple implication the causal relationship between A and B are not modelled appropriately.
  - Causality is reduced to logical equivalence when abstracting from timing. If we include timing the asymmetry between A and B becomes observable.
  - There exist times t such that $A(P{\downarrow}t)$ holds but not $B(P{\downarrow}t)$. We get for all times t (recall that we assume that A and B are stable)

    $B(P{\downarrow}t) \Rightarrow A(P{\downarrow}t)$

    $A(P{\downarrow}t) \Rightarrow \exists\, t' : t \leq t' \wedge B(P{\downarrow}t)$

Causality in systems

- Causality addresses the logical dependencies between the events (and in turn the actions) of a system.
- Causality is a way to "understand" a system.
  - ◇ Certain events and actions may take place or not (such as input). Other events and actions can be caused by these events and actions (such as output).
  - ◇ If a system has no rules of causality it is chaotic.
  - ◇ Then all actions and events may occur at all times in a completely unrelated random manner.

Causality: safety and liveness properties of systems

- An event A may guarantee an event B to happen later. This relationship is what is called a *liveness property*.
- An event B may only happen if event A has happened before. This relationship is what is called a *safety property*.
- There are basically two aspects of causality between two events (or actions) A and B. If A is causal for B, we may assume that
  - A **enforces** (**leads_to**) B: this means that whenever event A occurs event B eventually occurs (later),
  - B **requires** A: this means that event B does not occur if event A did not take place before.

Example. AirBag (continued)

- For the airbag we obviously get the rules

  crash_sensor_indicates_crash **enforces** airbag_is_activated

  as well as

  airbag_is_acivated **requires** crash_sensor_indicates_crash.

Causality and system properties

- A property Q is called *causal* for a property Q' if for each system run
  - ◇ whenever we observe Q at some time t at some time later t' > t we observe Q'.
  - ◇ Whenever we observe Q' at some time t' there exists some time with t < t' such that Q holds.
- We say "observation A is *causal* for observation B in system S" if for each process P ∈ S we have (for all times t ∈ TIME)

$$A(P{\downarrow}t) \Rightarrow \exists\, t' \in \text{TIME}: t \leq t' \wedge B(P{\downarrow}t')$$

$$B(P{\downarrow}t) \Rightarrow \exists\, t' \in \text{TIME}: t' \leq t \wedge A(P{\downarrow}t')$$

- The first formula is capturing what is called the "leads-to"-property in temporal logic expressed by temporal logic as follows:

$$\Box\,(A \Rightarrow \Diamond\, B)$$

The very nature of causality

- If we assume in systems that there are mechanisms ("algorithms") that enforce that certain actions are executed by some control flow, this gives us a very concrete ("operational") idea of causality.
  - On the other hand we may develop some idea about causality only by observing systems and their generated actions and try to conclude causality relations from that.

The very nature of causality

- Actually there are along these lines two ways to look at the causality of a system:
  - ◇ in an *intentional* view ("glass box view") we study the internal structure and mechanisms of a system to recognize that certain events A are causal for certain events B due to the technical mechanisms (the "algorithmics") that control the behavior and the flow of actions and events of the system.
  - ◇ in an *extensional* view ("black box view") to causality we study only observations about a system in terms of its events and actions and their temporal relationships without any knowledge of the internal structure and mechanisms of the system. In the set of all observations we may recognize certain regularities that we then call causal dependencies.
- In the first case we can speak about the causality within a single process or a single instance.
- In the second case we speak about the causality within the system considering all its processes.

Extensional causality

- Extensional causality can be seen as an abstraction of intentional causality.
  - ◊ By this abstraction some intentional causality may get hidden.
  - ◊ We get a universal notion of extensional ("observable") causality for a system S modeled by its set of timed processes.

Observing Causality

- If we can only observe runs of a system with the events and their timing we get processes that are sets of events with their timing.
  - From the set of all observations we cannot derive the intentional causality, in general.
  - Intentional causality allows us to talk about the causality within a single execution represented by a process, while extensional causality considers a system as a whole with all its processes.
- In spite of the example above, causality and nondeterminism are independent notions. We do not even have to have a notion of nondeterminism to introduce the concept of causality and vice versa.

Causality and Time

- Of course, we can also model causality for event structures that model systems by concurrent traces.
  - There causality is a logical relationship between the events of a system.
  - We work with a universal notion of extensional causality for a system S consisting of a set of timed processes.
- Time or at least a time-based ordering of the events are an indispensable instrument to capture and measure causality.
  - If event A is causal for event B then it certainly holds that B happens after A.
  - The reverse does not hold.
  - If in an observation an event A is before an event B this does not mean necessarily that they are causally related.
  - However, if A occurs in all processes (all observations) before B (whenever A occurs) then we may assume a causal relationship between the events A and B.

Strong and Weak Causality

- We call a system model strongly causal if all events that are intentionally causal to each other are strictly separated by their timing; more precisely the time scale is fine enough that whenever an event e1 is causal for an event e2 then we have

  $$time(e1) < time(e2)$$

  - Of course, in any case, we assume for causal events

    $$time(e1) \leq time(e2)$$

  - We speak of weak causality if at least this property holds for a system model.
  - Weak causality is what we at least expect for any faithful model of real world system. Only if we would include "incorrect" observations then the "law of weak causality" does not hold any longer.

Time Granularity and Causality

- Given a timed process P with the event set E and the timing function

$$\text{time: } E \rightarrow \text{TIME}$$

We can change ("transform") the timing of the process P by a function

$$\text{trans: TIME} \rightarrow \text{TIME'}$$

where we assume the following monotonicity property for all times t1, t2 $\in$ TIME:

$$t1 \leq t2 \Rightarrow \text{trans}(t1) \leq \text{trans}(t2)$$

Time Granularity and Causality

- Given a time transformation function we get a new timing for process P by the function

$$time': E \rightarrow TIME'$$

specified by the formula (for all events $e \in E$)

$$time'(e) = trans(time(e))$$

- As a result of a time transformation, the new timing may be coarser.
  - ◇ Events e1 and e2 with the timing property time(e1) < time(e2) may become simultaneous events under time'.
  - ◇ In other words, we may get time'(e1) = time'(e2).
  - ◇ We speak of a *time coarsening* in this case.

Interface View: Causality between Input and Output

- We study a specific form of causality for systems in the following, namely the causality between their input and output events. We work under following hypothesis:
  - there is a canonical notion of extensional causality between input and output.
- This idea of causality captures the essential relationship between input and output.
  - For a system we assume that we have a clear notion of input, which means input events can be chosen arbitrarily by the environment (and must be accepted by the system).
  - The input events form a sub-process called the input process. The system may or must react to such input by output.
  - This manifests the fundamental principle of causality between input and output.
  - Of course, in addition, there may be a causal relationship between the output events observable, which manifests some additional logics of the system.

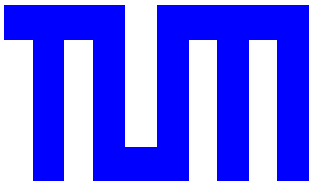
Hume and Causality

- Hume calls causality "the cement of the universe" -- implying that it holds everything together.
- Causality is an important topic in philosophy generally and in the philosophy of science.
- Understanding what
  - causes are helps us to understand how minds might (or might not) relate to bodies,
  - how free will might (or might not) work,
  - how ideas might (or might not) influence action, and
  - how bodies might come to produce changes in other bodies.

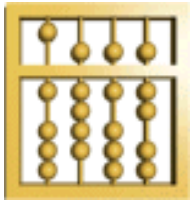
David Hume (April 26, 1711 – August 25, 1776), Scottish philosopher, economist, and historian is considered among the most important figures in the history of Western philosophy and the Scottish Enlightenment.

Relating Time and Causality in Interactive Distributed Systems

Manfred Broy



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



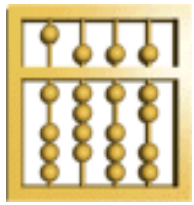
The Lectures I- IV

- Causality and Time in Discrete Systems
  - Discrete system models
  - Causality in systems
  - The role of time in system modeling
- A Modular System Model including Time and Causality
  - System interface behaviors
  - Specification
  - Composition
  - Refinement
- Changing the Granularity of Time
  - A flexible model of time
  - Time abstraction
  - Rules for Composition and Refinement
  - Causal Fixpoints and Causal Loops
- Delay Calculus
  - Flexible Timing of Systems
  - Composition and Delay
  - Optimal Delay Profile

Lecture II
A Modular System Model with Time and Causality

Manfred Broy

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



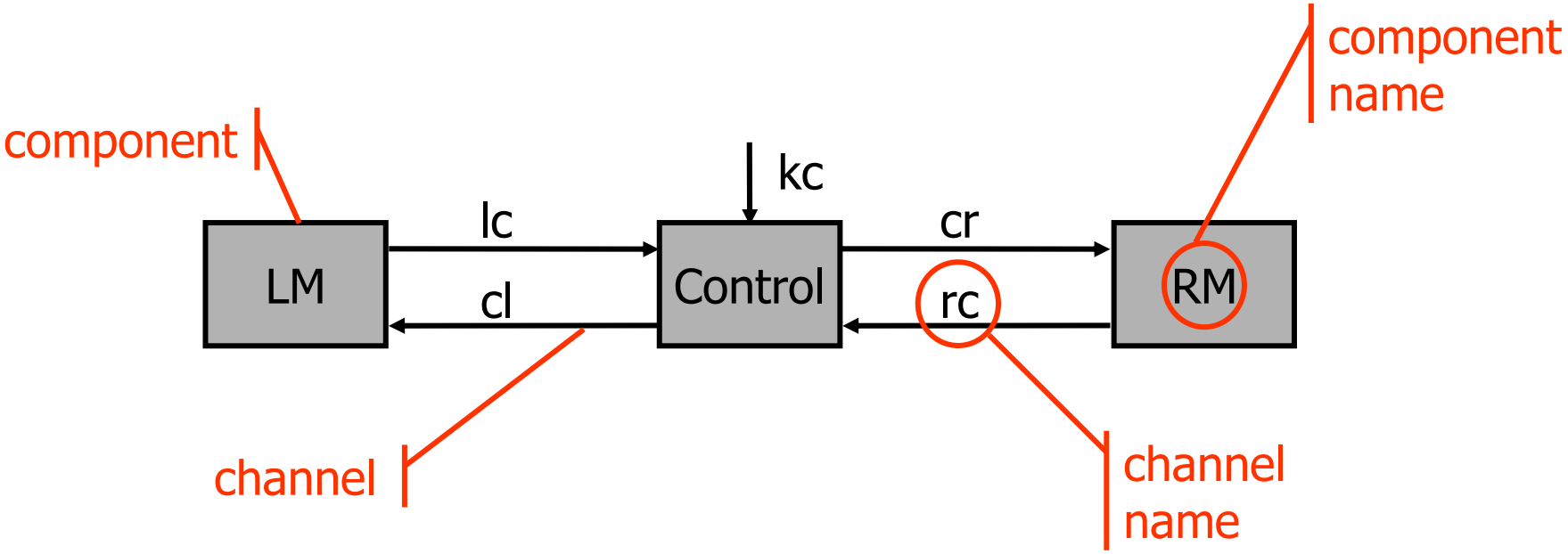
Interactive Asynchronous Systems with Interfaces

An interactive asynchronous system

- exchanges messages (input and output) with its environment via its input and output channels
- it has an interface which consists of its sets of input and output channels where for each channel a set of messages is specified
- the message exchange is carried out in a global time frame
- is an encapsulation of state (and corresponds to a state machine with input and output)

Towards a uniform model: Basic system model

System class: distributed, reactive systems



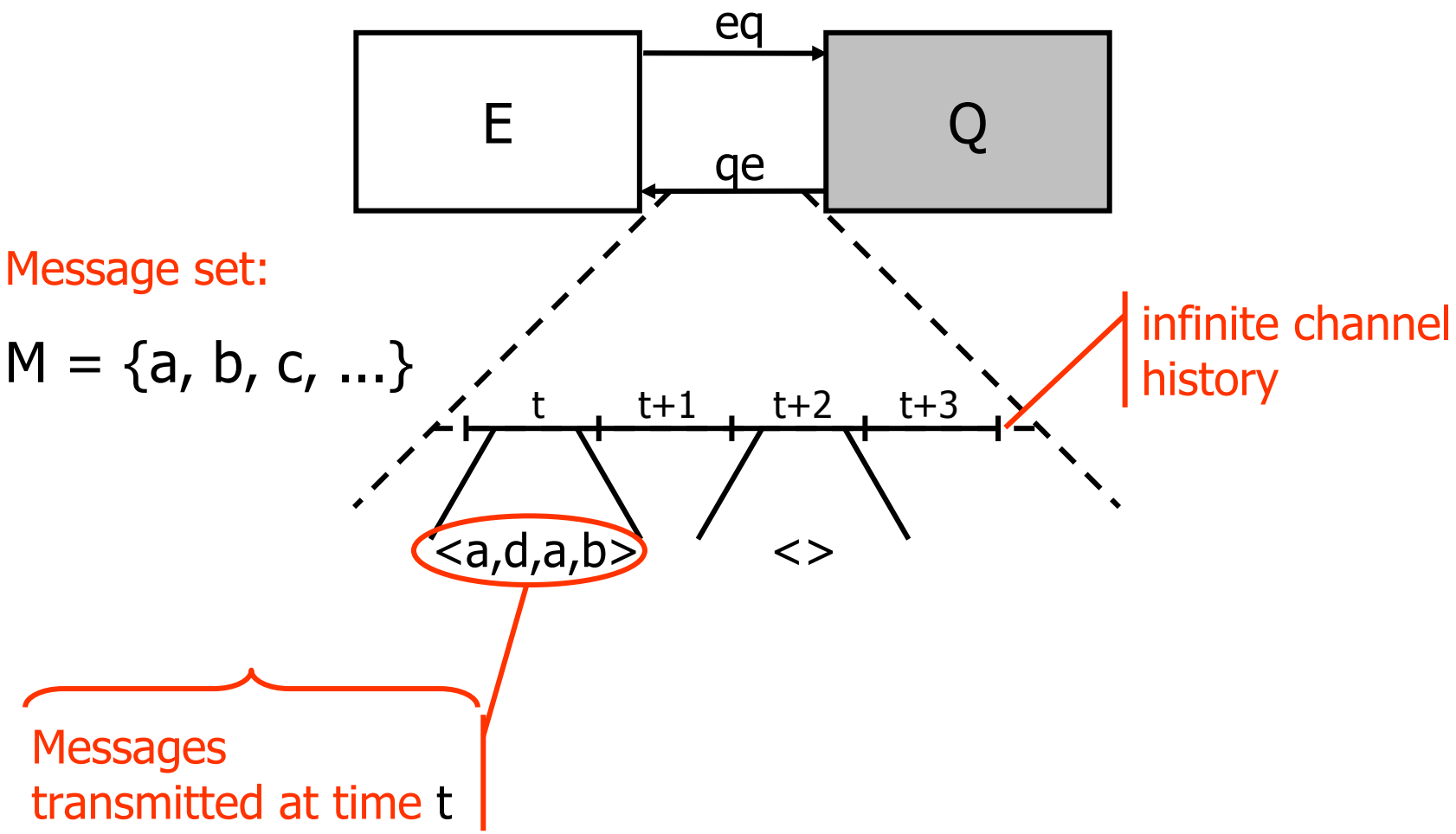
System consists of

- named components (with local state)
- named channels

driven by global, discrete clock

Basic Model

Timed Streams: Semantic Model for Black-Box-Behavior



Streams

- A stream of a given set M is a finite or an infinite sequence elements from M

  M^* denotes the set of finite sequences over M including the *empty* sequence $\langle\rangle$,

  M^∞ denotes the set of infinite sequences over M (that are represented by the total mappings $\mathbb{N}\backslash\{0\} \to M$).

- A stream is a member of the set M^ω that is defined by the equation

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

Streams - Notation

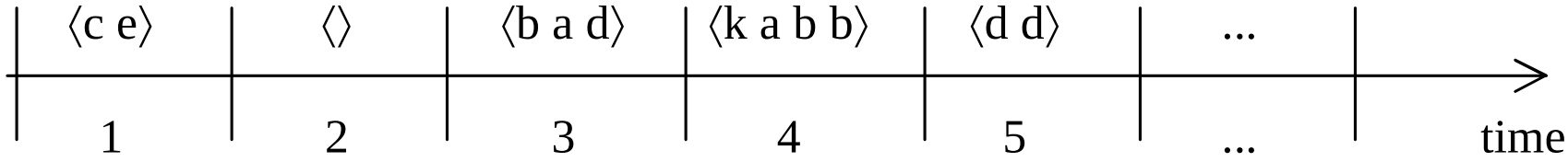
$\langle\rangle$	empty sequence or empty stream,
$\langle m \rangle$	one-element sequence containing m as its only element,
x.t	t-th element of the stream x, which is a sequence in the case of a timed stream
#x	length of the stream x,
x^z	concatenation of the sequence x to the sequence or stream z,
$x\downarrow t$	prefix of length t of the stream x (which is a sequence with t elements, in the case of a timed stream a sequence with t sequences as elements), provided x has at least t elements (otherwise $x\downarrow t = x$),
$S\square x$	stream obtained from x by deleting all its messages that are not elements of the set S.

Timed Streams

- A timed stream over the set M is given by an infinite stream a sequences of messages from M

$$(M^*)^\infty = \mathrm{IN}\backslash\{0\} \to M^*$$

An example of a timed stream over the set {a, b, c, ..., z}



The Basic Behaviour Model: Streams and Functions

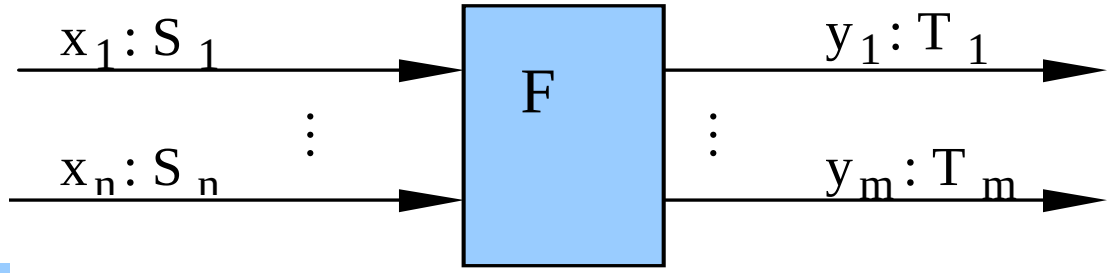
C	set of channels
$Type: C \to TYPE$	type assignment
$x : C \to (\mathbb{N} \setminus \{0\} \to M^*)$	channel history for messages of type M
$\vec{C}$ or $IH[C]$	set of channel histories for channels in C

System interface model

Channel: Identifier of Type stream

$I = \{ x_1, x_2, \ldots \}$ set of typed input channels
$O = \{ y_1, y_2, \ldots \}$ set of typed output channels

Interface behavior



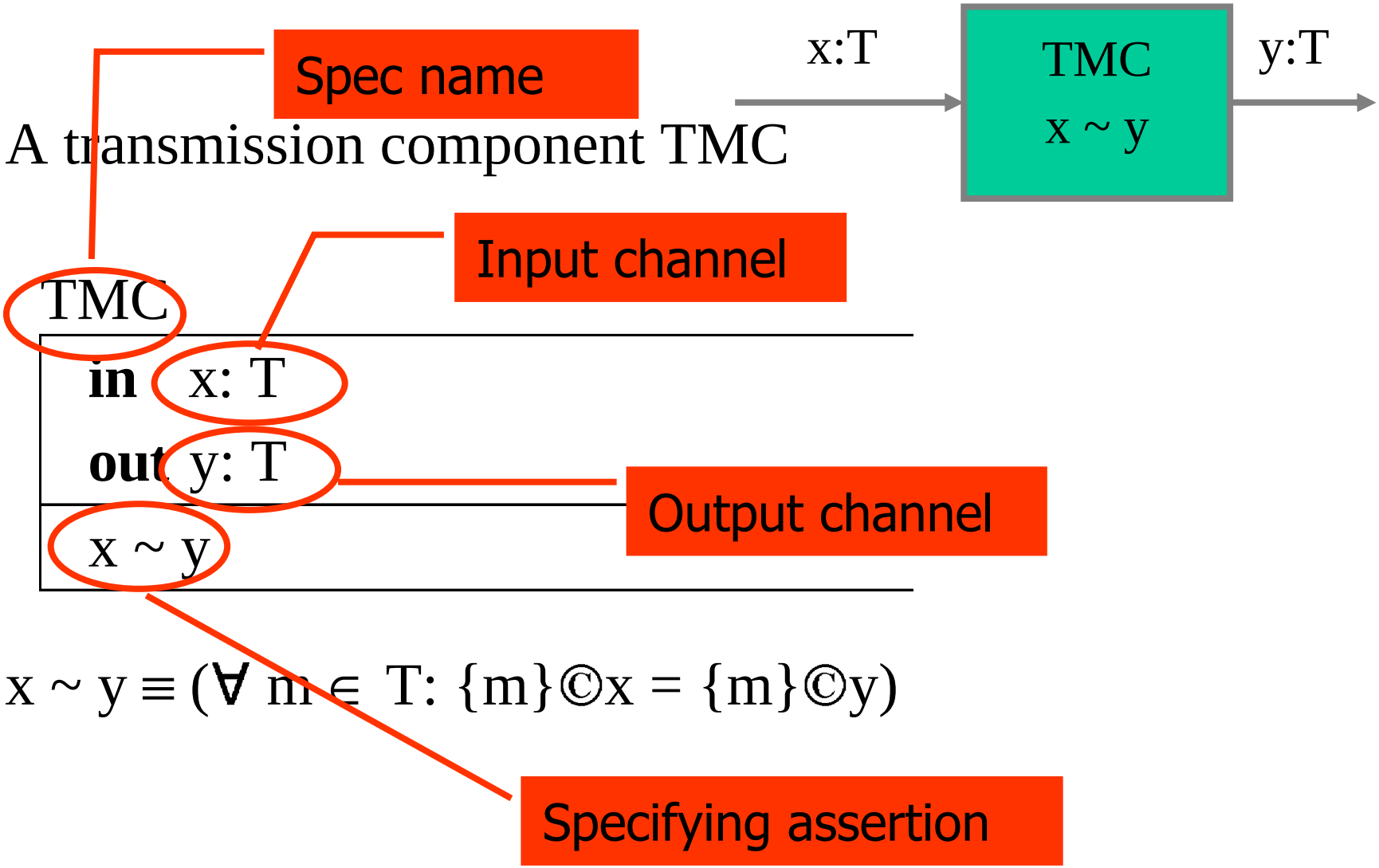
$$F : \vec{I} \rightarrow \wp(\dot{O})$$

Set of interfaces: IF[I ▶ O]

Specification of I/O-Behaviors

- An I/O-behavior represents a model of the behavior of a system.
- By logical means, an I/O-behavior F can be described by a logical formula, called *specifying assertion* relating the streams on the input channels to the streams on the output channels.
- In such a formula channel identifiers occur syntactically as identifiers (variables) for streams of the respective type.
- The specifying formulas are interpreted in the standard way of typed higher order predicate logic.
- An abstract specification of a system provides the following information:
  - its syntactic interface, describing the input and output channels by which the system interacts with its environment,
  - its behavior by a specifying formula Φ relating input and output channel valuations.

Example: Component interface specification



System interfaces and Causality

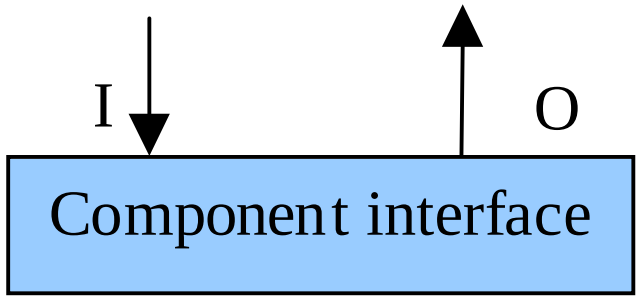
$(I \blacktriangleright O)$ *syntactic interface* with set of input channels I and of output channels O

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

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

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

A system is a total behavior

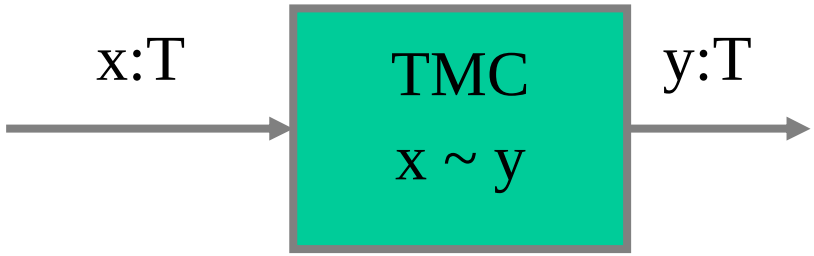


Causality

- Causality makes sure that there is a proper relationship between time and input/output
- Example: Causality for TCM

Channels	Interval 1	Interval 2	Interval 3	Interval 4	Interval 5
x	⟨a b b⟩	⟨ ⟩	⟨c b⟩	⟨d b e⟩	⟨ a d ⟩
y	⟨ ⟩	⟨b a⟩	⟨ b ⟩	⟨ c ⟩	⟨ ⟩

Example: TMC with Causality



TMC

in x: T **out** y: T
x ~ y $\wedge \forall t \in IN: \forall m \in T: \{m\}©(x\downarrow t) \sqsubseteq \{m\} ©(y\downarrow t+1)$

Delay by n Time Units

- We write delay(F, n), if F is a behavior with a delay by (at least) n time units:

$$delay(F, n) \equiv [\forall\, x, z, t: x{\downarrow}t = z{\downarrow}t \Rightarrow (F.x){\downarrow}t+n = (F.z){\downarrow}t+n]$$

- In other words,
  - ◇ F is (weakly) causal if delay(F, 0) holds and
  - ◇ strongly causal if delay(F, 1) holds.
- For all n, m $\in$ IN (the proof is straightforward):

$$n \leq m \wedge delay(F, m) \Rightarrow delay(F, n)$$

- There is a maximal number n such that delay(F, n) holds provided the input has effects on the output.

Deterministic behaviour

An I/O-behavior:

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

is isomorphic to a relation which is a subset of $I^r \times O^r$.

Given an input history $x \in I^r$, F.x denotes the set of all output histories that F may show in reaction to the input x.

F is called *deterministic*,
if F.x is a one-element set for each $x \in I^r$.

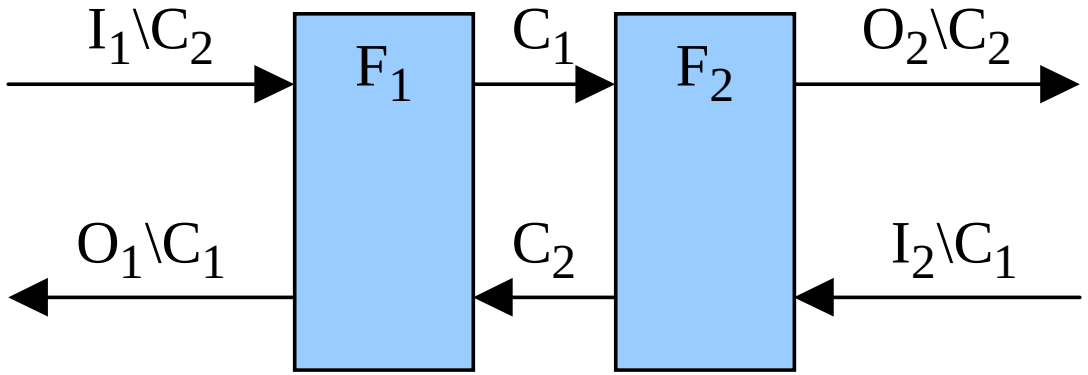
A deterministic I/O-behavior represents a function $f: I^r \to O^r$.

f is called a realization of a behavior F if $f.x \in F.x$ for all $x \in I^r$

Composition of Systems

$F_1 \in \mathbb{F}[I_1 \blacktriangleright O_1]$
$F_2 \in \mathbb{F}[I_2 \blacktriangleright O_2]$

$C_1 = O_1 \cap I_2$
$C_2 = O_2 \cap I_1$
$I = I_1 \backslash C_2 \cup I_2 \backslash C_1$
$O = O_1 \backslash C_1 \cup O_2 \backslash C_2$



$F_1 \otimes F_2 \in \mathbb{F}[I \blacktriangleright O]$,

$(F_1 \otimes F_2).x = \{z|O: x = z|I \wedge z|O_1 \in F_1(z|I_1) \wedge z|O_2 \in F_2(z|I_2)\}$

Composition of Deterministic Systems

$f_1 \in \mathbb{F}[I_1 \blacktriangleright O_1]$ deterministic

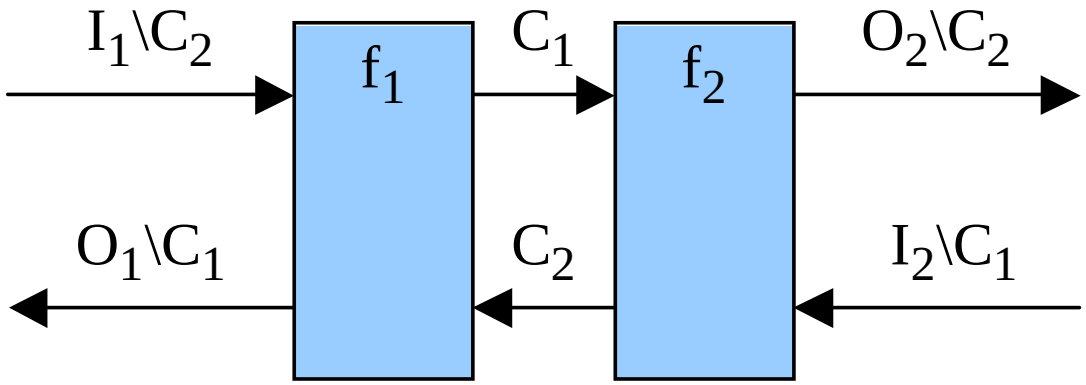
$f_2 \in \mathbb{F}[I_2 \blacktriangleright O_2]$ deterministic

$C_1 = O_1 \cap I_2$

$C_2 = O_2 \cap I_1$

$I = I_1 \backslash C_2 \cup I_2 \backslash C_1$

$O = O_1 \backslash C_1 \cup O_2 \backslash C_2$



$f_1 \otimes f_2 \in \mathbb{F}[I \blacktriangleright O]$,

$(f_1 \otimes f_2).x = \{z|O: x = z|I \wedge z|O_1 = f_1(z|I_1) \wedge z|O_2 = f_2(z|I_2)\}$

This is a fixpoint definition

Composition of Specifications into Architectures

Input channels

Output channels

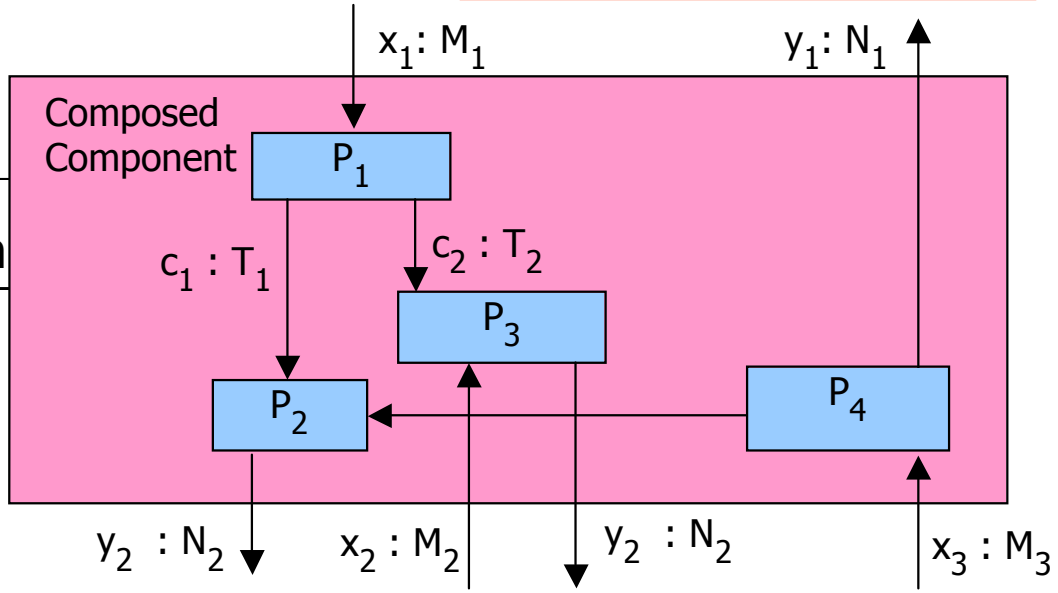
Internal channels

Composed component spec

in $x_1: M_1 , x_2: M_2 , \ldots$

out $y_1: N_1 , y_2: N_2 , \ldots$

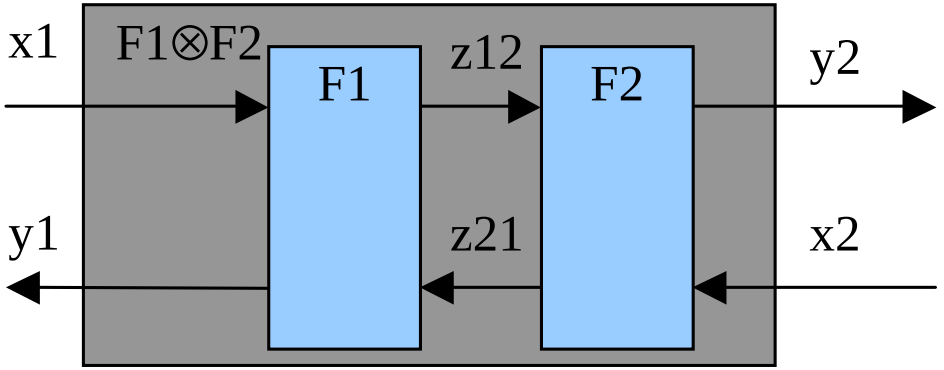
$\exists\, c_1, c_2 , \ldots : P_1 \wedge \ldots \wedge P_n$



System composition = logical und

Channel Hiding = existential quantification

Interface specification composition rule



F1

in x1, z21: T
out y1, z12: T
P1

F2

in x2, z12: T
out y2, z21: T
P2

F1⊗F2

in x1, x2: T
out y1, y2: T
∃ z12, z21: P1 ∧ P2

Composition for Strongly Causal Behaviors

- For deterministic strongly causal behaviors f_k for k = 1, 2

  $f_1 \otimes f_2$ is a strongly causal deterministic behavior

  - the prove uses an inductive construction and uses the existence of fixpoints
- If f_k are realisations of F_k for k = 1, 2, then

  $f_1 \otimes f_2$ is a realisation of $F_1 \otimes F_2$

  - every history $y \in F_1 \otimes F_2$ has a computational justification (is causally justified)

Composition for Weakly Causal Behaviors

- For deterministic weakly causal behaviors f_k for k = 1, 2 it is not guaranteed
  - ◇ that $f_1 \otimes f_2$ is a deterministic behavior
  - ◇ that there exists outputs $y \in (f_1 \otimes f_2)(x)$ at all for all input histories x
  - ◇ that outputs $y \in (f_1 \otimes f_2)(x)$ correspond to causal computations (we speak of causal loops)

  since
  - ◇ for deterministic weakly causal behaviors fixpoints are neither unique nor guaranteed to exist
- If weakly causal behaviors f_k are realisations of F_k for $k = 1, 2$, then it is not guaranteed that

$$f_1 \otimes f_2 \text{ is a realisation of } F_1 \otimes F_2$$

Grandfather paradox

- The grandfather paradox is a paradox of time travel, first described by the science fiction writer René Barjavel in his 1943 book Le Voyageur Imprudent (The Imprudent Traveller).
  - The paradox is this: suppose a man traveled back in time and killed his biological grandfather before the latter met the traveller's grandmother.
  - As a result, one of the traveller's parents (and by extension, the traveller himself) would never have been conceived.
  - This would imply that he could not have travelled back in time after all, which in turn implies the grandfather would still be alive, and the traveller would have been conceived, allowing him to travel back in time and kill his grandfather.
  - Thus each possibility seems to imply its own negation, a type of logical paradox.
- An equivalent paradox is known (in philosophy) as autoinfanticide — that is, going back in time and killing oneself as a baby — though when the word was first coined in a paper by Paul Horwich it was in the version autofanticide
- The grandfather paradox has been used to argue that backwards time travel must be impossible.

Fixpoints of strongly causal functions

For strongly causal deterministic behavior

$$f: \vec{I} \to I^r$$

we calculate stepwise inductively a fixpoint $y \in I^r$.

We start with t = 0; by definition y↓0 is empty. Given y↓t with y↓t = (f.y)↓t we get the sequence y.t+1 = (f.y')↓t for arbitrary y' $\in I^r$ with y↓t = y'↓t.

Note that by strong causality y↓t+1 = (f.y)↓t+1.

By construction y = f.y.

By construction y is a fixpoint and unique.

Fixpoints and weakly causal functions

For a weakly causal deterministic behavior

$$f: \vec{I} \to I^{r}$$

fixpoints are

1) not guaranteed (may not exist)

   $f: (IN^*)^\infty \to (IN^*)^\infty$

   $(f.x).t = succ^*(f.x)$ where $succ.\langle a_1 \ldots a_n \rangle = \langle a_1+1 \ldots a_n+1 \rangle$

2) are not unique (there may be several ones)

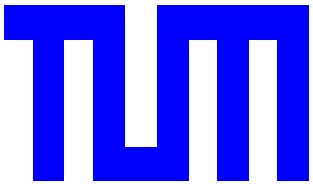
   Example: $f.x = x$

Advantages of strong causality

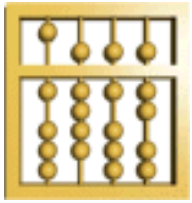
- Composition of data flow nodes corresponds to a fixpoint construction (in the presence of feedback loops)
- Strong causality makes the composition construction correct
  - ◇ Each fixpoint corresponds to a computation
  - ◇ Every computation corresponds to a fixpoint
- No causal loops! (see Lecture III)

Relating Time and Causality in Interactive Distributed Systems

Manfred Broy



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

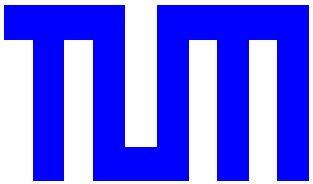


The Lectures I- IV

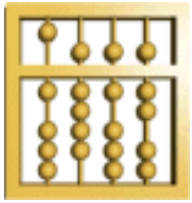
- Causality and Time in Discrete Systems
  - Discrete system models
  - Causality in systems
  - The role of time in system modeling
- A Modular System Model including Time and Causality
  - System interface behaviors
  - Specification
  - Composition
  - Refinement
- Changing the Granularity of Time
  - A flexible model of time
  - Time abstraction
  - Rules for Composition and Refinement
  - Causal Fixpoints and Causal Loops
- Delay Calculus
  - Flexible Timing of Systems
  - Composition and Delay
  - Optimal Delay Profile

Lecture III
Changing the Granularity of Time

Manfred Broy



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

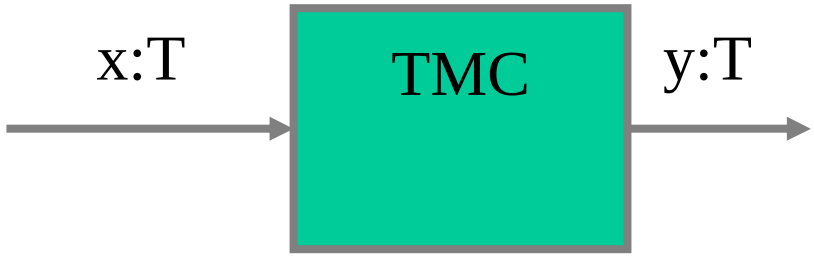


A flexible model of time

Time is a key issue in embedded systems:

- Dealing with timing properties
  - ◇ Specification
  - ◇ Analysis
  - ◇ Verification
    - Analysis
    - Testing
    - Model checking
    - Deduction based verification
- Transforming time
- Dedicated models of time
  - ◇ Micro/Macro Step
  - ◇ Perfect synchrony
  - ◇ Scheduling
- Abstractions

Example: TMC with Timing Restrictions



TMC

in x: T **out** y: T
∀ t ∈ IN: ∀ m ∈ T: {m}©(x↓t) ▯ {m} ©(y↓t+delay) {m}©(x↓t) Š {m}©(y↓t+delay+deadline)

Making the time scale coarser for a history

Let $n \in \mathbb{N}$; to make for a history (or a stream)

$$x \in \vec{C}$$

the time scale coarser by the factor n we use the function

$$COA(n): \vec{C} \to \wp(\vec{C})$$

defined by $t \in \mathbb{N}$:

$$COA(n)(x).t+1 = x.n*t+1\hat{}...\hat{}x.n*t+n$$

Making the time scale coarser for a behavior

To make for a behavior

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

the time scalecoarser by the factor n, we define the coarsening operator

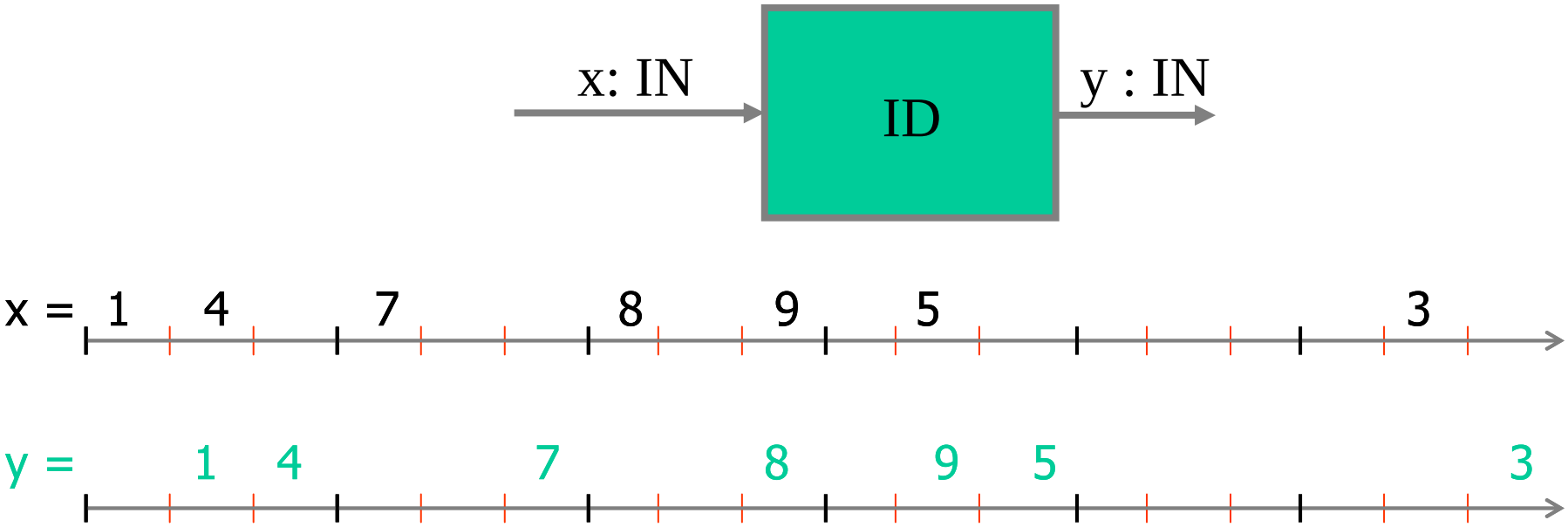
$$COA(F, n): \dot{I} \to \wp(\dot{O})$$

as follows

$$COA(F, n)(x) = \{ COA(n)(y): \exists\, x': x = COA(n)(x') \wedge y \in F(x') \}$$

On histories, coarsening is a function that is not injective and thus there does not exist an inverse.

Making time granularity coarser



We get nondeterminism in the timing of the output

we loose nondeterminism due to distinctions in the input

Time abstraction

A coarsening factor n = ∞ in time coarsening mapps the infinite number of time intervals in a timed stream is mapped into one. The infinite stream of sequences is concatenated into a nontimed stream.

$$COA(\Box).x = \overline{x}$$

Making the Time Scale Finer

Let $n \in \mathbb{N}$; to make for a history (or a stream)

$$x \in \vec{C}$$

its time scale finer by the factor n we use the function

$$\text{FINE}(n): C^r \to \wp(C^r)$$

defined by the equation:

$$\text{FINE}(n)(x) = \{x' \in C^r : \forall\, t \in \mathbb{N}: x.t+1 = x'.(n*t+1)^\frown \ldots ^\frown x'.(n*t+n)\}$$

FINE(n).x yields the set of histories where for each time interval the sequences of messages in this interval are arbitrarily subdivided into n sequences that are associated with n successive time intervals.

Making the Time Scale Finer

Another way to define the function FINE is demonstrated by the following formula

$$FINE(n)(x) = \{x' : COA(n).x' = x\}$$

To make a time scale finer for behaviors we specify

$$FINE(F, n)(x) =$$

$$\{y' \in FINE(n).y : \exists\, x' : x = FINE(n)(x') \wedge y \in F(x')\}$$

Rules for Time Scale Refinement

Rules for changing the time scale show that the functions COA(n) and FINE(n) form refinement pairs in the sense of granularity refinement:

$$COA(n).FINE(n).x = \{x\}$$

$$x \in FINE(n).COA(n).x$$

Time Scale Refinement as Granularity Refinement

The following equations hold:

$$COA(F, n) = FINE(n) \circ F \circ COA(n)$$

$$FINE(F, n) = COA(n) \circ F \circ FINE(n)$$

here $F_1 \circ F_2$ denotes the pipeline composition of

F_1: $\vec{I}_1 \to \wp(O_1^r)$ and F_2: $I_2^r \to \wp(O_2^r)$

where $O_1 = I_2$ and

$$(F_1 \circ F_2).x = \{y: \exists z: z \in F_1.x \wedge y \in F_2.z\})$$

Property Refinement of Interfaces

Property refinement allows us to replace an interface behavior by one having additional properties. An interface

$$F: \vec{I} \to \wp(\dot{O}) \text{ is refined by } \hat{F}: \dot{I} \to \wp(\dot{O})$$

if

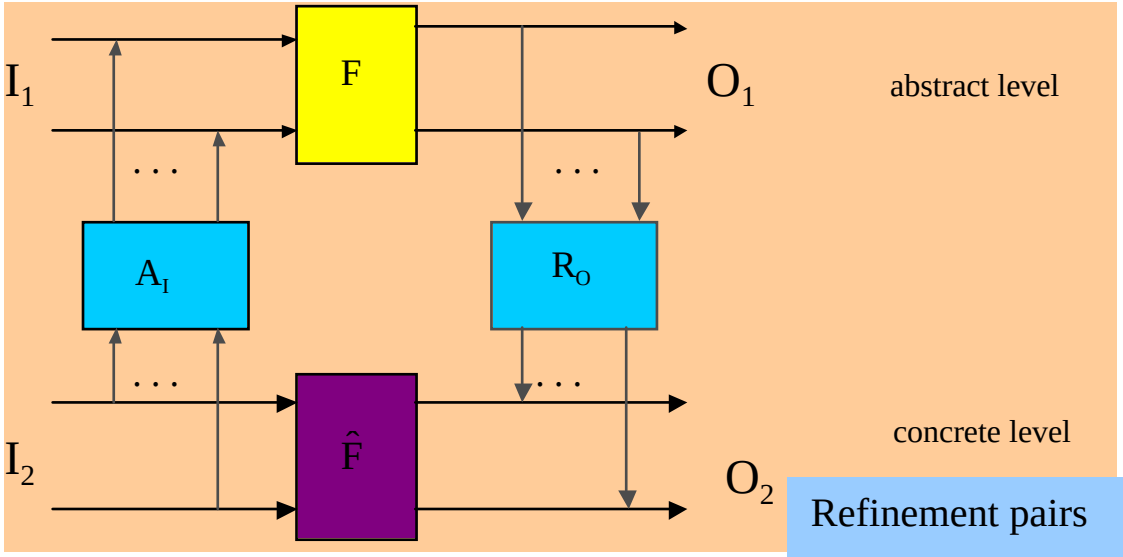
$$\forall x \in \dot{I}: \hat{F}(x) \subseteq F(x)$$

We write then

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

Obviously, property refinement is a partial order and therefore reflexive, asymmetric, and transitive.

Granularity Refinement: Changing Levels of Abstraction



Refinement pairs

$A_I : \vec{I}_2 \to \wp(\dot{I}_1)$ $\quad R_I : \dot{I}_1 \to \wp(\dot{I}_2)$ $\quad R_I \mathbin{;} A_I = Id$

$A_O : \dot{O}_2 \to \wp(\dot{O}_1)$ $\quad R_O : \dot{O}_1 \to \wp(\dot{O}_2)$ $\quad R_O \mathbin{;} A_O = Id$

specify a relationship between streams

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

is called granularity refinement

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

if

$\hat{F} \subseteq A_I \mathbin{;} F \mathbin{;} R_O$ $\qquad$ *U^{-1}-simulation*

Rules for Composition and Refinement

We get the following obvious rules:

$$FINE(n*m) = FINE(n) \circ FINE(m)$$

$$COA(n*m) = COA(n) \circ COA(m)$$

and the following equation (let $m, n \in \mathbb{N}$ with $m \geq 1$):

$$delay(F, n*m) \Rightarrow delay(COA(F, m), n)$$

The proof of this rule is quite straightforward and uses the fact that

$$(COA(m).x)\downarrow t = COA(m).(x\downarrow(t*m))$$

holds.

Rules for Delay of Coarsened Behaviours

Monotonicity of coarsening with respect to delay properties:

$$n < m \wedge \text{delay}(\text{COA}(F, m), k) \Rightarrow \text{delay}(\text{COA}(F, n), k)$$

Fractions of Time Changes

We can even change in a behavior the time by fractions m/n for $m, n \in \mathbb{N}$ using the time change operator TCH.

$$TCH(F, m/n) = COA(FINE(F, n), m)$$

Note that all the rules introduced so far for COA and FINE carry obviously over to this case, since we expressed TCH in terms of COA and FINE.

Causal Fixpoints and Causal Loops

Given a strongly causal function

$$F: C^{r} \rightarrow C^{r}$$

each fixpoint $y \in F.y$ is called causally faithful or for short *causal*.

€ €

In a causal fixpoint each sequence of values in a time interval t is triggered by the values that occurred in the history before t.

Causal Fixpoints and Causal Loops

Consider a coarsening of the time for the function F

$$COA(F, n): \ C^{r} \rightarrow C^{r}$$

We get for each fixpoint

$$y \in F.y \in$$

that COA(n).y is a fixpoint of COA(F, n), too.

But there may be fixpoints COA(F, n) that do not correspond to fixpoints of F.

A fixpoint $y' \in (COA(F, n)).y'$ is called *causal w.r.t. F* if there is a fixpoint $y \in F.y$ such that $y' = COA(n).y$.

Otherwise y' is called a *causal loop*.

Causal Fixpoints and Causal Loops

- Considering COA(F, n) alone without knowing F we cannot distinguish, in general, causal fixpoints from causal loops.

Example: Causal Fixpoints and Causal Loops

IDS: $\vec{C} \to \wp(\dot{C})$

with the specification (for all channels $c \in C$)

$(IDS.x).c = \{ \langle\rangle^\frown(x.c) \}$

IDS is the identity on the data streams shifted by one time interval.

Example: Causal Fixpoints and Causal Loops

Consider the history

$$x.c = \langle\langle 1\rangle\ \langle\rangle\rangle^{\infty} \text{ for all } c \in C$$

we get

$$(IDS.x).c = \langle\langle\rangle\ \langle 1\rangle\rangle^{\infty} \text{ for all } c \in C$$

This shows that x is not a fixpoint of IDS.

Example: Causal Fixpoints and Causal Loops

For COA(2).IDS we get:

$$COA(2).\langle\langle 1\rangle\ \langle\rangle\rangle^{\infty} = \{\ \langle\langle 1\rangle\rangle^{\infty}\ \}$$

$$COA(2).\langle\langle\rangle\ \langle 1\rangle\rangle^{\infty} = \{\ \langle\langle 1\rangle\rangle^{\infty}\ \}$$

This proves the fixpoint property

$$\langle\langle 1\rangle\rangle^{\infty} \in (COA(2).IDS).\langle\langle 1\rangle\rangle^{\infty}$$

So $\langle\langle 1\rangle\rangle^{\infty}$ is a fixpoint of COA(2).IDS.

However, this fixpoint is not causal, since $\langle\langle\rangle\rangle^{\infty}$ is the only fixpoint of IDS as well as COA(IDS, 2).

Coarsing time and causal fixpoints

- This shows that coarsening a behavior F leads to functions F′ with additional fixpoints
  - ◇ that are not causal in the case that F′ is, in contrast to F, not strongly but only weakly causal.
  - ◇ Moreover, we cannot distinguish, in general, for these functions causal from non-causal fixpoints.

Choosing the Appropriate Time Scale

- We establish and model different time scales for the subsystems of a composed system.
- Then we can choose the time scales in a flexible way, according to the following observations:
  - for each system composed of strongly causal components its time delay is greater than the length of the shortest path of channels through the system of components.
  - therefore we can coarsen the interface abstraction of the system by the factor k without loosing strong causality provided the shortest path is ≥ k.
- This leads to hierarchical system models that support local islands ("subsystems") of finer granularity of time.
- A system may be composed of many subsystems with their own finer time scales.

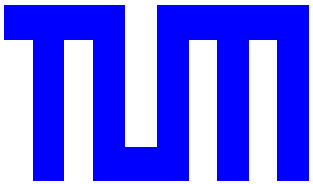
Hume and Causality

- Hume calls causality "the cement of the universe" -- implying that it holds everything together.
- Causality is an important topic in philosophy generally and in the philosophy of science.
- Understanding what
  - causes are helps us to understand how minds might (or might not) relate to bodies,
  - how free will might (or might not) work,
  - how ideas might (or might not) influence action, and
  - how bodies might come to produce changes in other bodies.

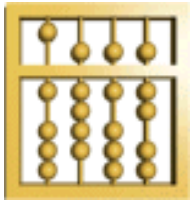
David Hume (April 26, 1711 – August 25, 1776), Scottish philosopher, economist, and historian is considered among the most important figures in the history of Western philosophy and the Scottish Enlightenment.

Relating Time and Causality in Interactive Distributed Systems

Manfred Broy



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

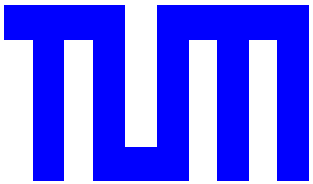


The Lectures I- IV

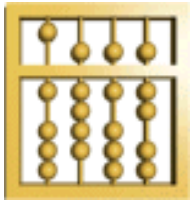
- Causality and Time in Discrete Systems
  - Discrete system models
  - Causality in systems
  - The role of time in system modeling
- A Modular System Model including Time and Causality
  - System interface behaviors
  - Specification
  - Composition
  - Refinement
- Changing the Granularity of Time
  - A flexible model of time
  - Time abstraction
  - Rules for Composition and Refinement
  - Causal Fixpoints and Causal Loops
- Delay Calculus
  - Composition and Delay
  - Optimal Delay Profile
  - Reasoning with and without Causality
  - Flexible Timing of Systems

Lecture VI
Delay Calculus

Manfred Broy

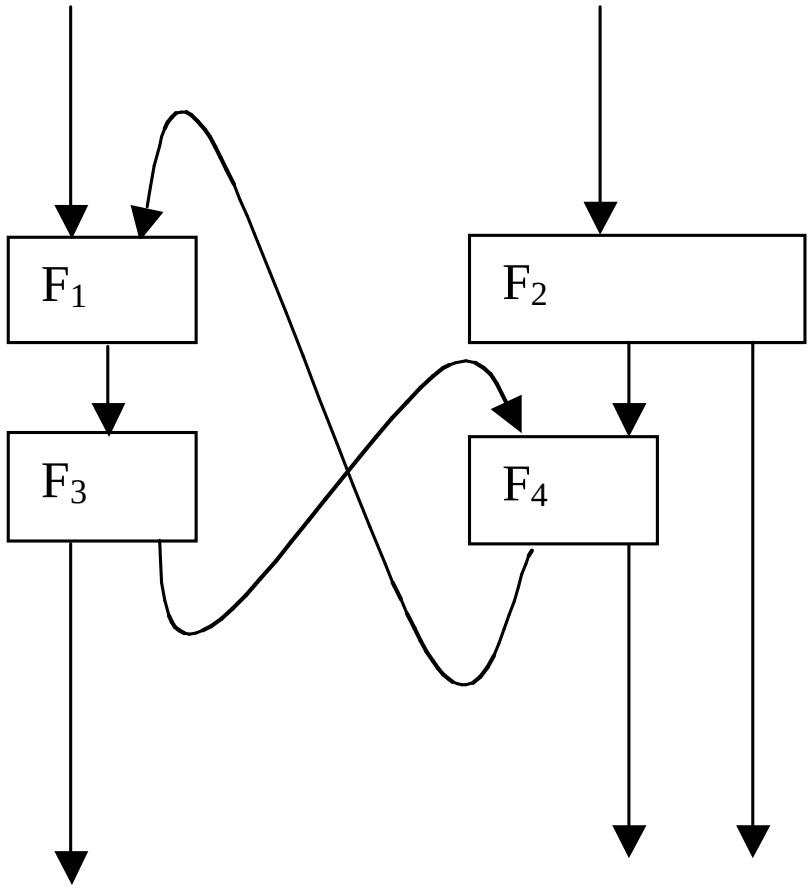


Technische Universität München
Institut für Informatik
D-80290 München, Germany



Delay Calculus for System Architectures

- By composition of a family of components we can form a network (representing an architecture) of components with delays.



Delay Calculus for System Architectures

- This network is a directed graph with channels as arcs and components as nodes.
- With each path in the graph leading from an input channel to an output channel c we can associate a delay which is the sum of all delays on that path (each component on the path adds to the delays).
- For an output channel c the (guaranteed) delay in the system for c is the minimum over the sum of (guaranteed) delays on each of the paths from some input channel to the output channel c.

Delay Calculus for System Architectures

In a behavior function we can define the guaranteed delay between the input channels and an output channel c. Consider

$$F: I^r \rightarrow \wp(O^r)$$

and an output channel $c \in O$. We define the guaranteed delay for the output channel c in F by the following formula:

$$\in \quad \in$$

$$\text{gardelay}(F, c) =$$

$$\max \{k \in \mathbb{IN} \cup \{\infty\}: \forall\, x, x', t \in \mathbb{IN}:$$

$$x{\downarrow}t = x'{\downarrow}t \Rightarrow ((F.x).c){\downarrow}t+k = ((F.x').c){\downarrow}t+k\}$$

Delay length of a path from input channel c to output channel c′

A communication path p through a network is a sequence of channels

$$p = \langle c_0\ c_1\ \ldots\ c_k \rangle$$

such that for each index $i < k$ there exists

> a component F_i in the system such that c_i is an input channel of F_i and c_{i+1} is an output channel of that component.

c_0 is called the source of p and c_k is called the target of p.

By $Path(c, c')$ we denote the set of all paths with channel c as its source and c'channel as its target.

Delay length of a path from input to output channel c

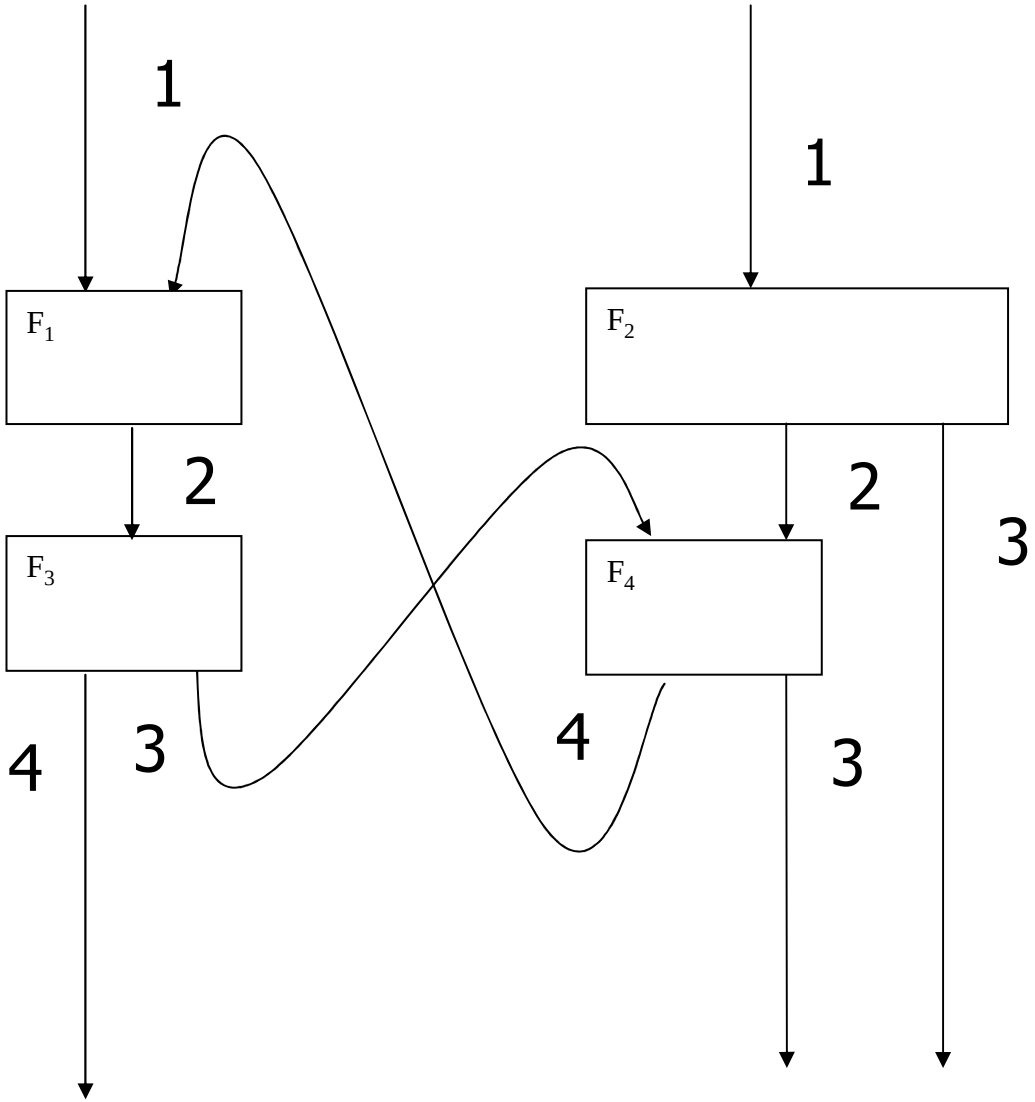
The delay length of the path is given by the formula

$$dl(p) = \sum_{i=1}^{k} \text{gardelay}\ (F_i, c_i)$$

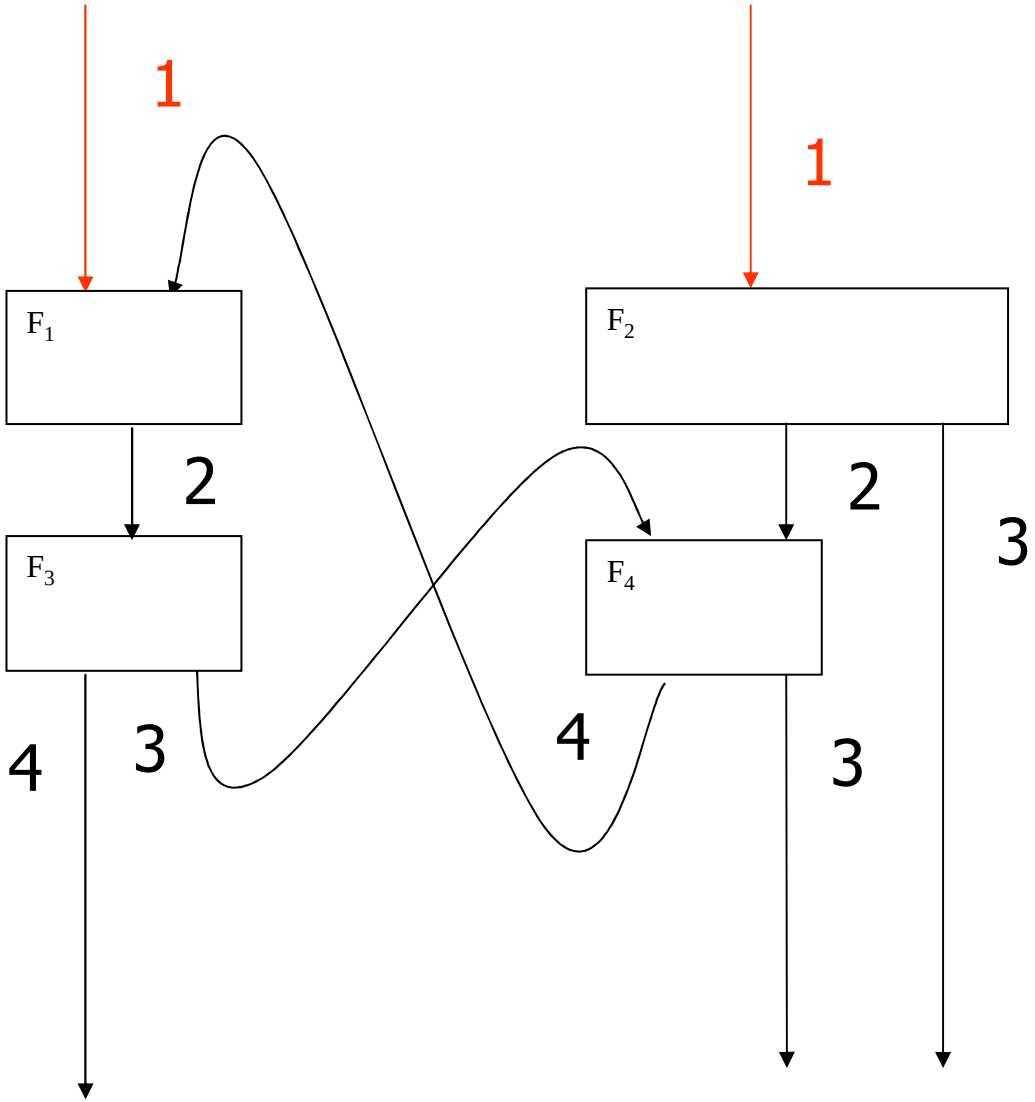
For each output channel c of the network for the system F we obtain:

$$\text{gardelay}(F, c) \geq \min \{dl(p): \exists\ b \in I: p \in \text{Path}(b, c)\ \}$$

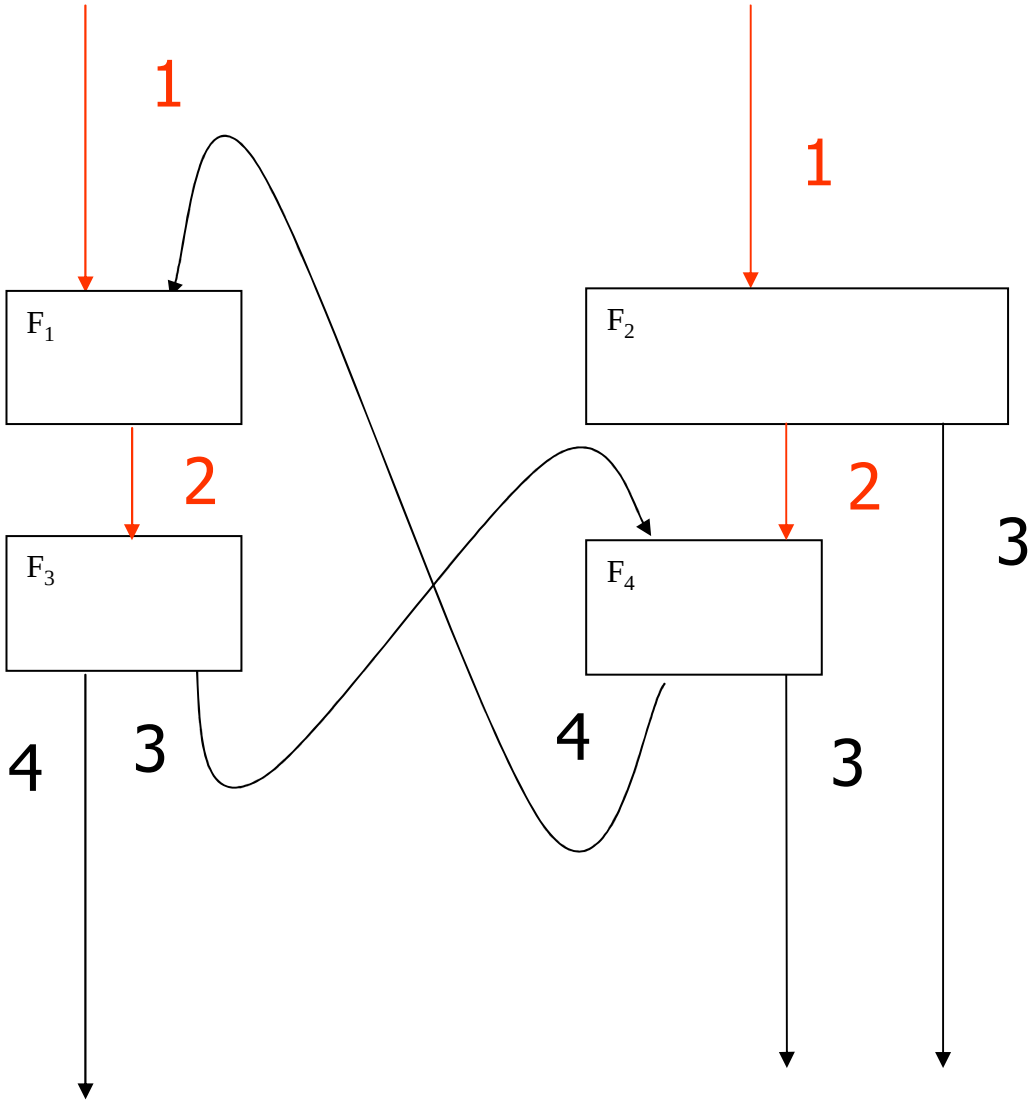
The flow of time: step 0



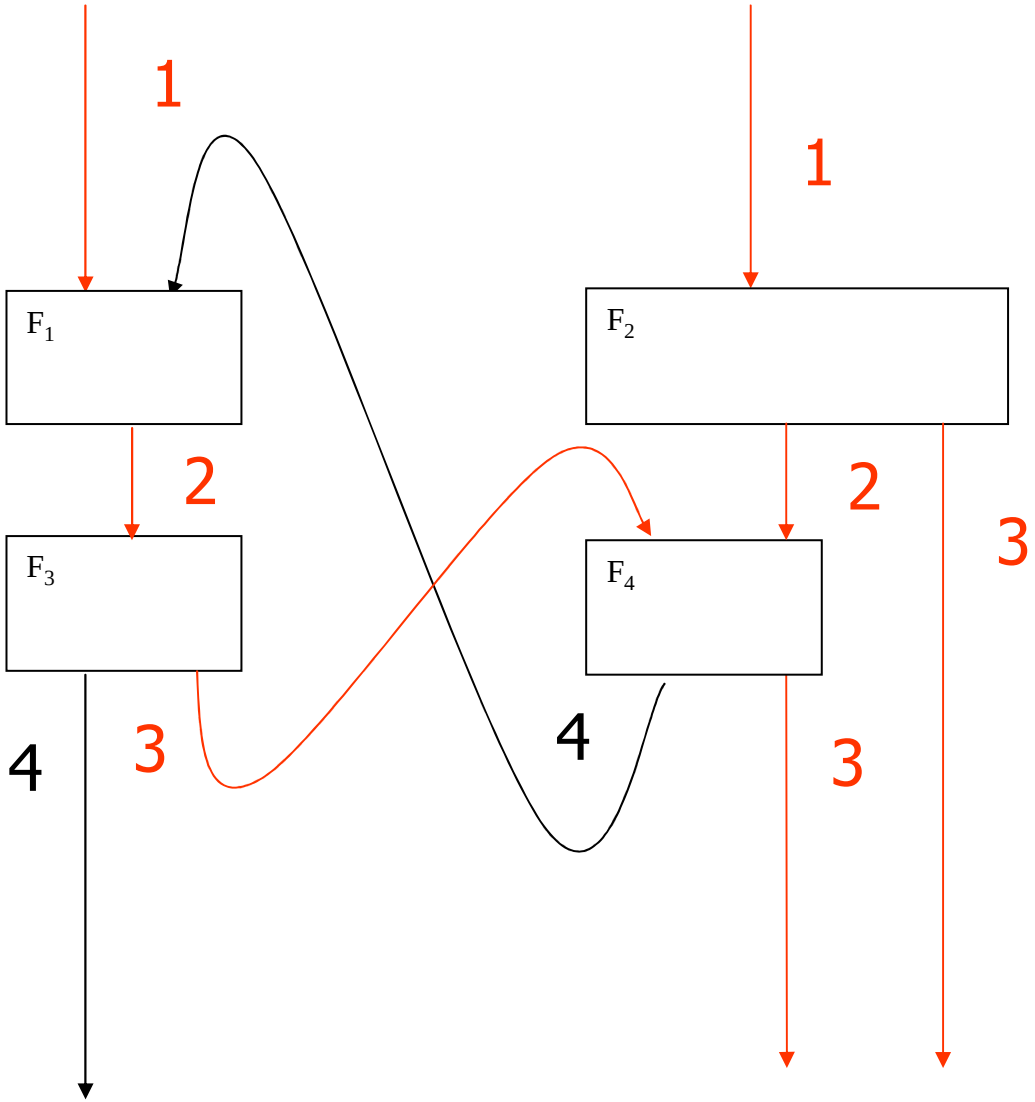
The flow of time: step 1



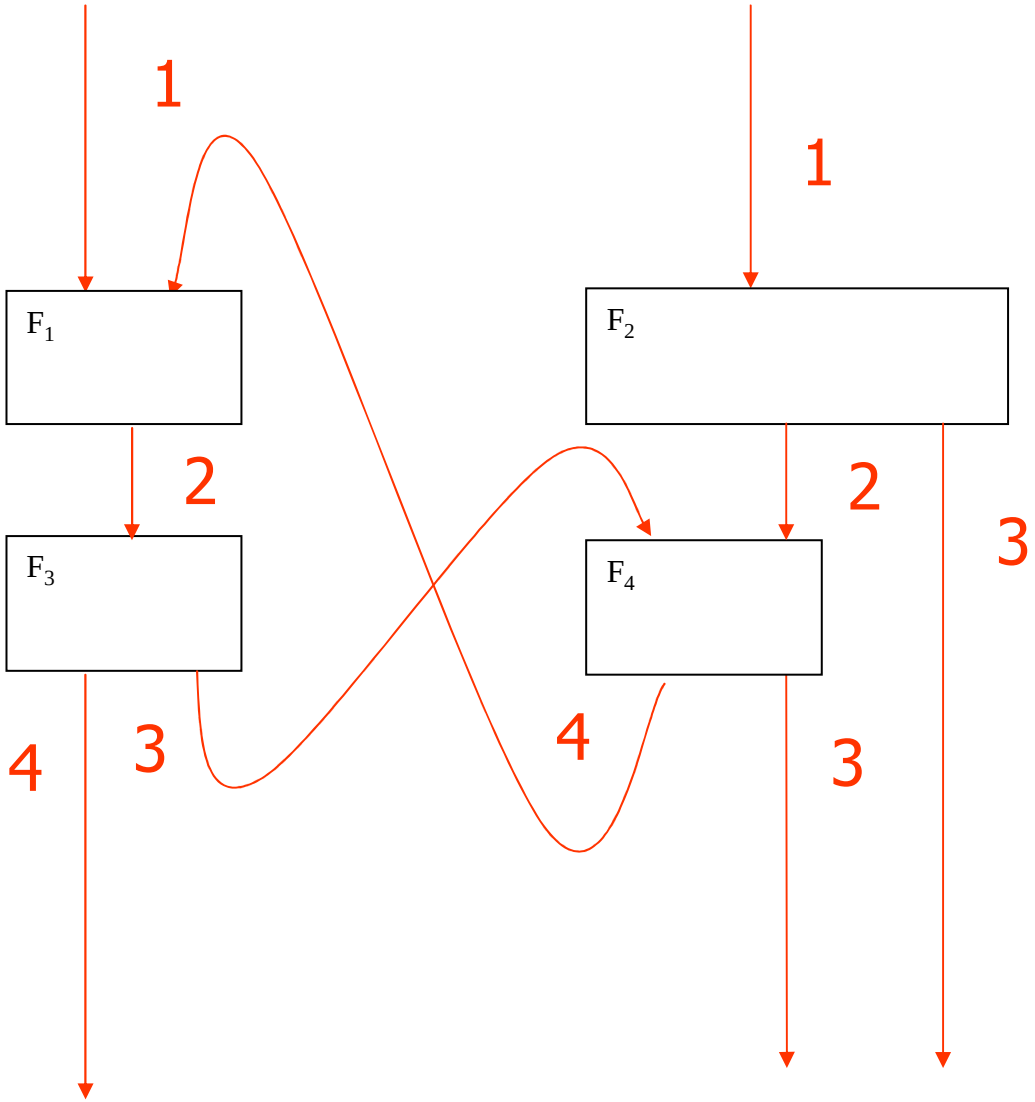
The flow of time: step 2



The flow of time: step 3



The flow of time: step 4



Composition and Delay

For each component of the considered system, we can define a (maximal) guaranteed delay for each output channel.

Given a component

$$F: I^r \to \wp(O^r)$$

we introduce mappings

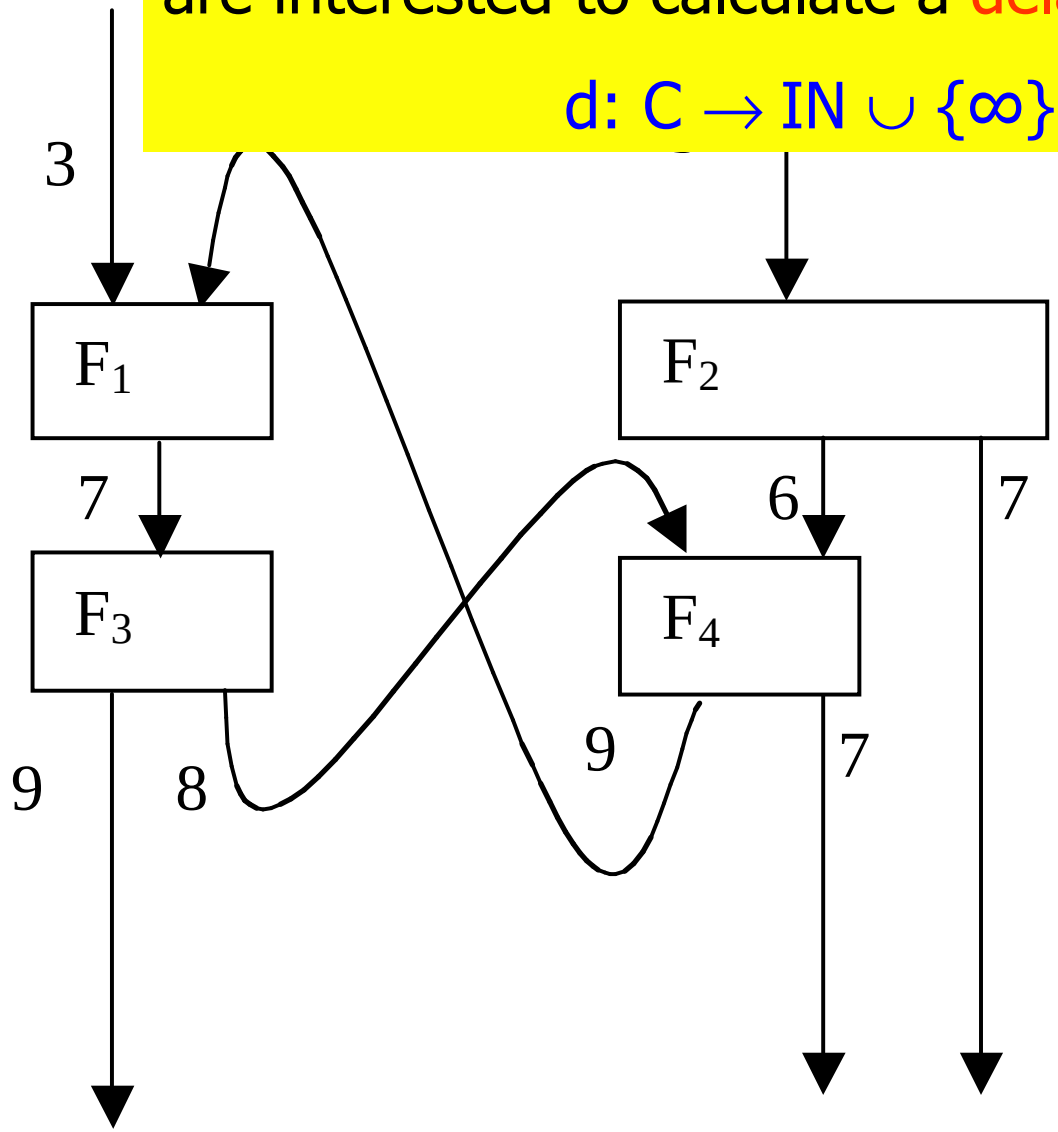
$$d: I \to \mathbb{N} \cup \{\Box\}, e: O \to \mathbb{N} \cup \{\Box\}$$

that associate a delay with every input and output channel.

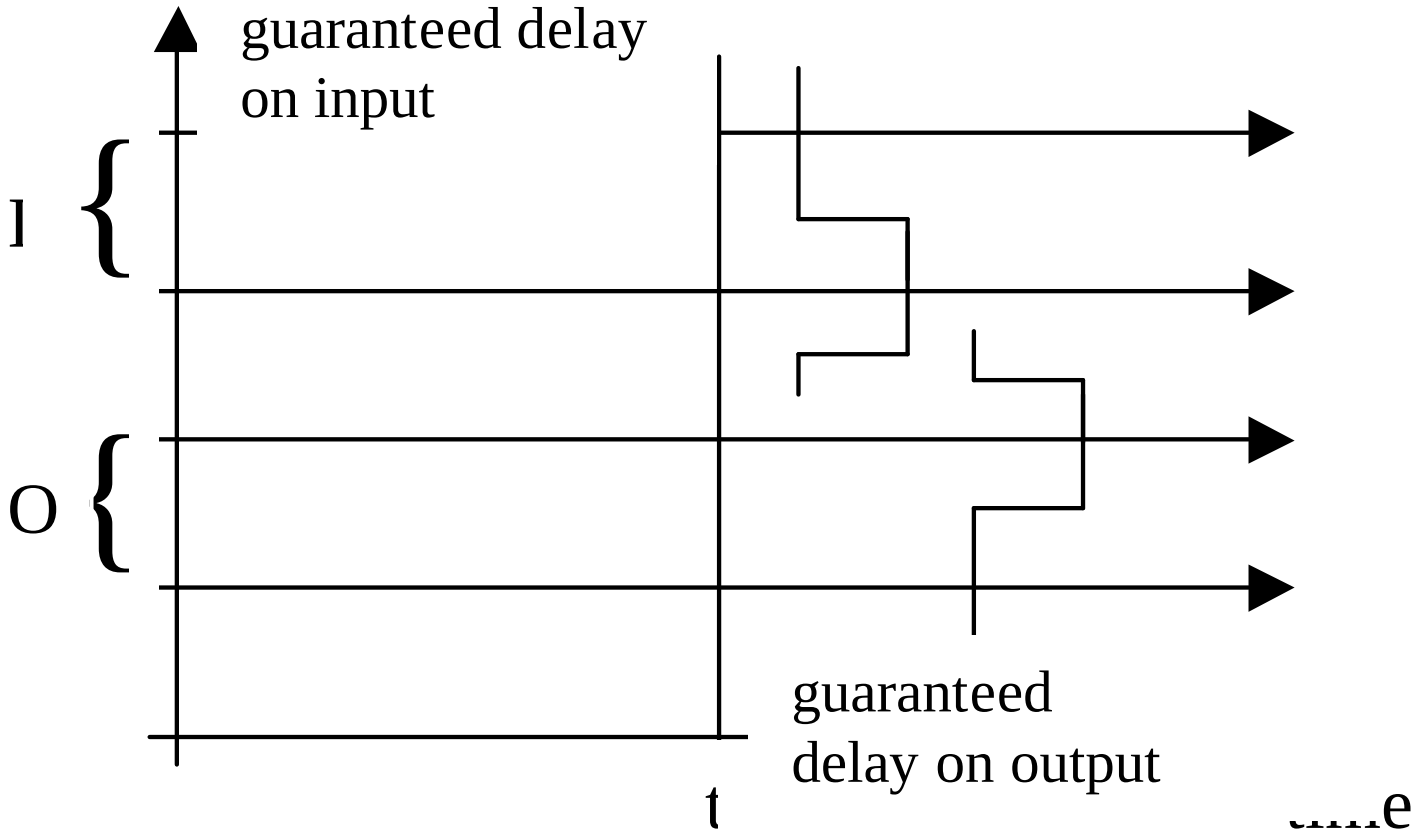
To calculate the guaranteed delay of a network of components, we have to be able to determine the delay of F if for each of its input channels a delay is given.

Let C be a set of channels in the network. We are interested to calculate a delay profile

$$d: C \to IN \cup \{\infty\}$$

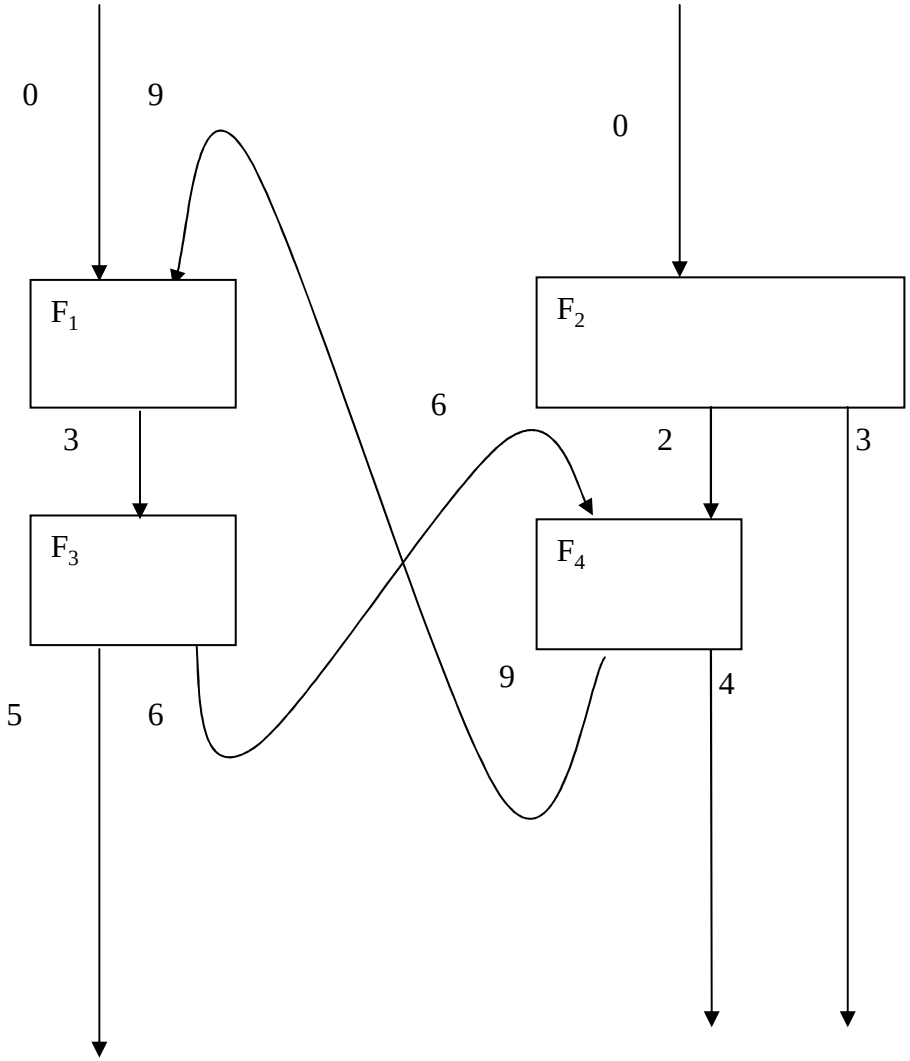


Composition and Delay

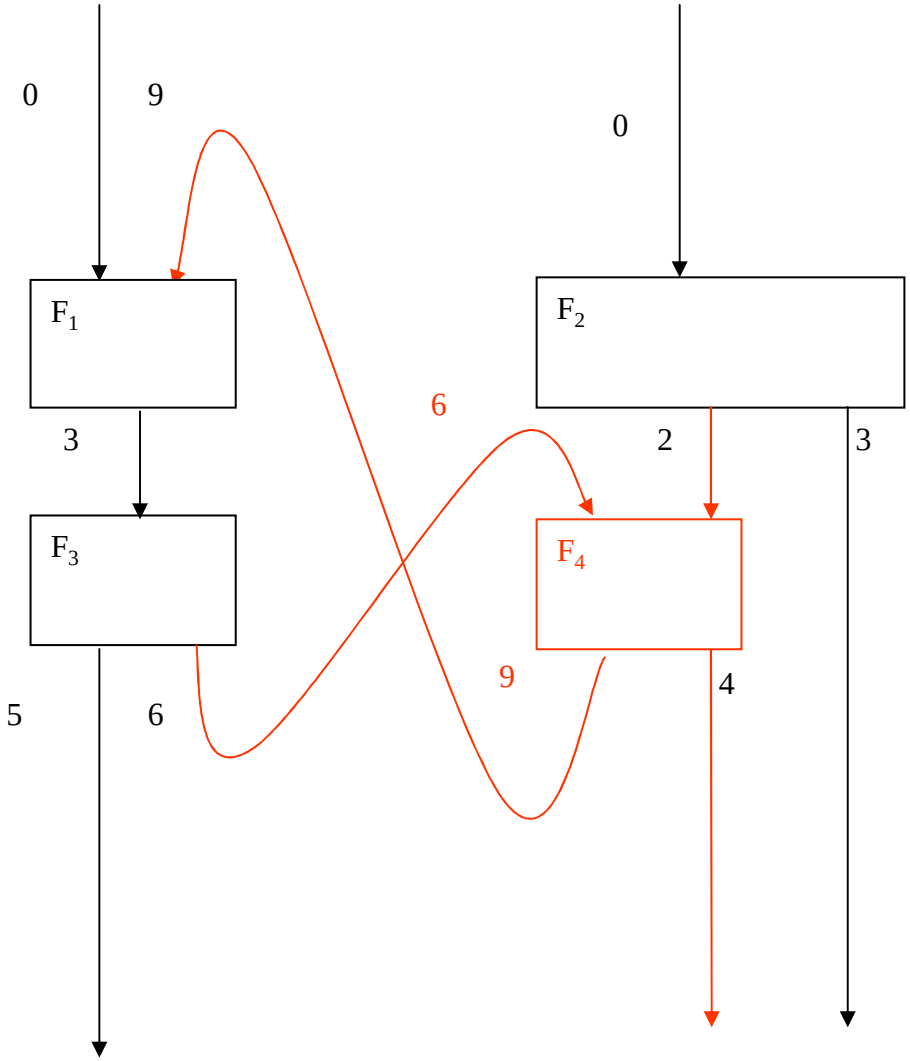


Relationship between input with guaranteed delay and output with guaranteed delay at time t

Delay Profile



Delay Profile



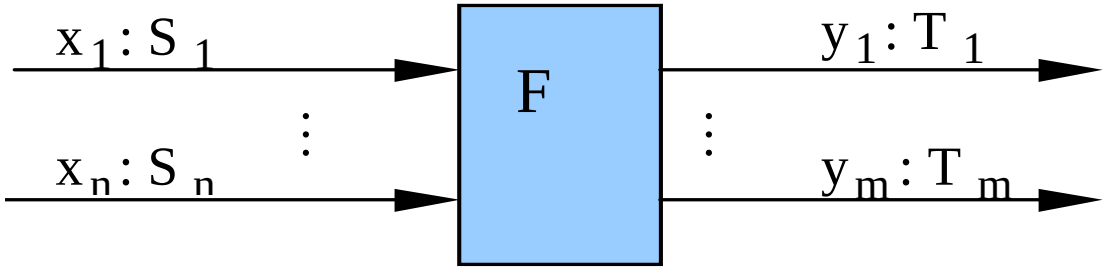
A System of Equations to Calculate Delay

- With every system behavior F we may associate a function

$$gdel_F(c): IN \times ... \times IN \rightarrow IN$$

for every output channel c

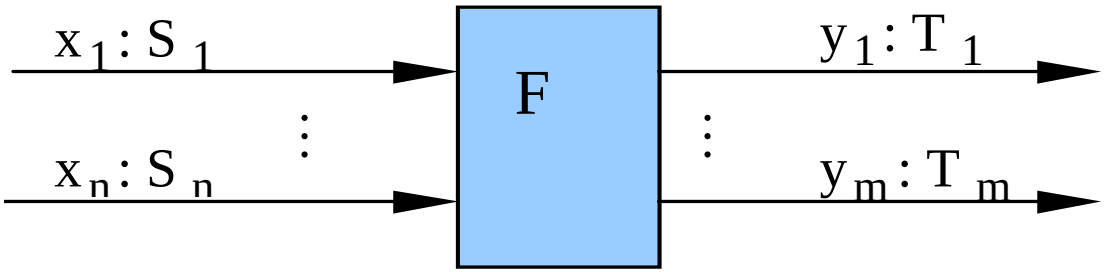
defined by



$$gdel_F(c)(d.c_1, ..., d.c_n) =$$
$$\max \{k \in IN \cup \{\infty\} : \forall\ x, z:$$
$$[\forall\ b \in I: (x.b)\downarrow(t-d.b) = (z.b)\downarrow(t-d.b)] \Rightarrow$$
$$\{(y.c)\downarrow t+k : y \in F.x\} = \{(y.c)\downarrow t+k : y \in F.z\}$$

From networks to equations

- Given a network with nodes



- We associate with every node and every output channel c of one of the nodes F the equation

$$d.c = gdel_F(c)(d.x_1, ..., d.x_n)$$

  to calculate for for every channel c in the network its delay d.c.
- For the input channels c we assume d.c = 0.

Delay equations

- This gives a set of equations from a network, one for each channel $c \in C$

$d.c = gdel_F(c)(d.x_1, ..., d.x_n)$ if c has a system F as its source

$d.c = 0$ if c is input channel to the network

where

$$d: C \to IN \cup \{\infty\}$$

- We choose the least fixpoint for d to get the maximal guaranteed delays.

Composition and the Choice of the Time Scale

- We study the question how time abstraction and composition fit together.
- A compositional formula for time refinement should read as follows:

$$COA(F_1 \otimes F_2, n) = COA(F_1, n) \otimes COA(F_2, n)$$

- However, this formula does not hold, in general, since making a behavior coarser is an information loss that may result in the loss of strong causality and thus may introduce "causal loops".
  - The time abstraction is the origin of the problems with causal loops in approaches advertised under the name "perfect synchrony" such as Esterel.
  - Moreover, the individual timing of the subcomponents may be highly relevant for selecting the behaviors (the output histories).

Composition and the Choice of the Time Scale

Theorem: Let F_1 and F_2 be behaviors that can be composed (according to consistent channel naming and typing); then

$$COA(F_1 \otimes F_2, n) \subseteq COA(F_1, n) \otimes COA(F_2, n)$$

The theorem shows that

$$COA(F_1 \otimes F_2, n)$$

is a refinement of

$$COA(F_1, n) \otimes COA(F_2, n)$$

The converse statement is not true, in general.

- It holds only if both F_1 and F_2 are
  - ◊ strongly causal and
  - ◊ not sensitive to the finer timing.

Composition and the Choice of the Time Scale

Theorem:

Let $n > 1$ hold. Assume for $i = 1, 2$ that we have (for all x, x')

$$COA(n).x = COA(n).x' \Rightarrow F_i.x = F_i.x'$$

and that $delay(F_i, n)$ holds, then we get:

$$COA(F_1, n) \otimes COA(F_2, n) \subseteq COA(F_1 \otimes F_2, n)$$

Moreover, if a component is time independent, then we have:

$$COA(F, n).COA(n).x = COA(n).(F.x)$$

Finer Time Granularity

Trivially, the equation

$$\mathrm{FINE}(F_1 \otimes F_2, n) = \mathrm{FINE}(F_1, n) \otimes \mathrm{FINE}(F_2, n)$$

does always hold, as long as F_1 and F_2 are strongly causal.

Assumptions for Causality

- As formulated in the hypothesis above, we may assume
  - ◇ that for each model of a physical system behavior there is a time scale that is fine enough to capture all essential time differences especially for the delay between input and output to guarantee the property of strong causality.
  - ◇ Modeled in an appropriate time scale the behavior is always strongly causal according to the principle of strong causality.

Pros for Strong Causality

- Strong causality has a number of significant advantages since
  - ◇ it makes the reasoning about systems more concrete and simpler since reasoning about feedback loops is reduced to induction.
  - ◇ In particular, it is easy to treat feedback loops by fixpoints for strongly causal behaviors since strong causality guarantees, in particular, the existence of unique fixpoints for deterministic functions.
  - ◇ In other words, for strongly causal, fully realizable system behaviors all fixpoints are causal and thus computationally appropriate in the sense that they faithfully reflect computations.

Cons for Strong Causality

Let the behaviors

$$F_1: I_1 \to \wp(O_1),\ F_2: O_1 \to \wp(O_2)$$

be given. We obtain

$$\text{delay}(F_1, m) \wedge \text{delay}(F_2, n) \Rightarrow \text{delay}(F_1 \circ F_2, m+n)$$

In the case of two strongly causal functions F_1 and F_2 we get (at least)

$$\text{delay}(F_1 \circ F_2, 2)$$

On one hand this observation is very satisfactory since it leads to a useful delay calculus.

Cons for Strong Causality

- This it shows an unfortunate inflexibility of the design calculus for timed systems.
  - ◇ If we want to represent a function by an architecture with two functions composed by pipelining we always have to accept a delay by at least 2 if the functions are strongly causal.
  - ◇ In fact, if we insist on a delay less than 2 then a component cannot be implemented by a system consisting of two components composed sequentially.
  - ◇ This seems unacceptable, since it makes the design very inflexible, and seems to be a good reason to reject our approach based on a global discrete time altogether.

Avoiding Causal Loops

In the following we finally discuss the problem of causal loops. We show how we can add strong causality as a property to specifications to reason about feedback loops.

Example: Interface specification
Consider the following specification of a component that copies its input on channel x on both of its output channels y and r.

Copy

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

Strong causality as an assumption on the specification Copy

Causality and Composition

Given a set of specified components we get for each component F: $I \to \wp(O)$ with the relational specification ("specifying assertion") for $y \in F.x$ in the form of a predicate P:

$$P(x, y)$$

by imposing causality the logical weakest specification Pc that fulfills the following equation

$$Pc(x, y) \equiv P(x, y) \wedge \forall\, t \in \mathbb{N}: \exists\, x' : x'{\downarrow}t = x{\downarrow}t \Rightarrow$$
$$\exists\, y': Pc(x', y') \wedge y'{\downarrow}t+1 = y{\downarrow}t+1$$

This way we can add the requirement of strong causality to every specification to get a strongly causal specification.

Avoiding Causal Loops

Example: Interface specification

Strong causality as an assumption on the specification Copy allows us to conclude:

$$\forall\, t \in \mathbb{N}: \forall\, m \in T:$$

$$\{m\}\#\, \overline{r \downarrow t+1} \sqsubseteq \{m\}\#\, \overline{x \downarrow t}$$

$$\wedge \quad \{m\}\#\, \overline{y \downarrow t+1} \sqsubseteq \{m\}\#\, \overline{x \downarrow t}$$

where $M\#x$ denotes the number of copies of messages in the set that occur in x.

$\in$

Avoiding Causal Loops

Repeater

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

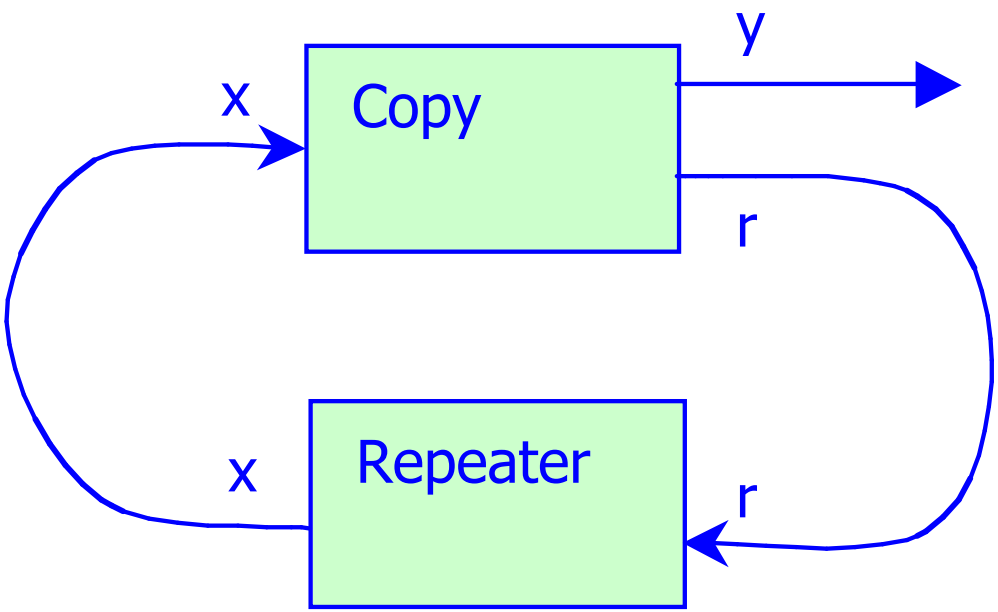
Strong causality allows us to conclude in addition to the specifying assertion the following property

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

Avoiding Causal Loops

We compose the two components

Repeater ⊗ Copy



Composition Repeater ⊗ Copy

Avoiding Causal Loops

Without strong causality arguments we get the following formula:

$$\bar{x} = \bar{y} \wedge \bar{x} = \bar{r}$$

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

$$\wedge \{m\}\#\bar{r} = 0 \Rightarrow \{m\}\#\bar{x} = 0$$

which simplifies to

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

which indicates for the output y of the network only

$$\forall m \in T: \{m\}\#y > 0 \Rightarrow \{m\}\#y = \infty$$

Avoiding Causal Loops

Without strong causality arguments we get the following formula:

$$\bar{x} = \bar{y} \wedge \bar{x} = \bar{r}$$

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

$$\wedge \{m\}\#\bar{r} = 0 \Rightarrow \{m\}\#\bar{x} = 0$$

With strong causality arguments we get in addition the assertion

$$\forall\, t \in \mathbb{N}: \forall\, m \in T:$$

$$\{m\}\#\overline{r\downarrow t+1} \sqsubset \{m\}\#\overline{x\downarrow t} \wedge \{m\}\#\overline{y\downarrow t+1} \sqsubset \{m\}\#\overline{x\downarrow t}$$

$$\wedge\, \forall\, t \in \mathbb{N}: \forall\, m \in T:$$

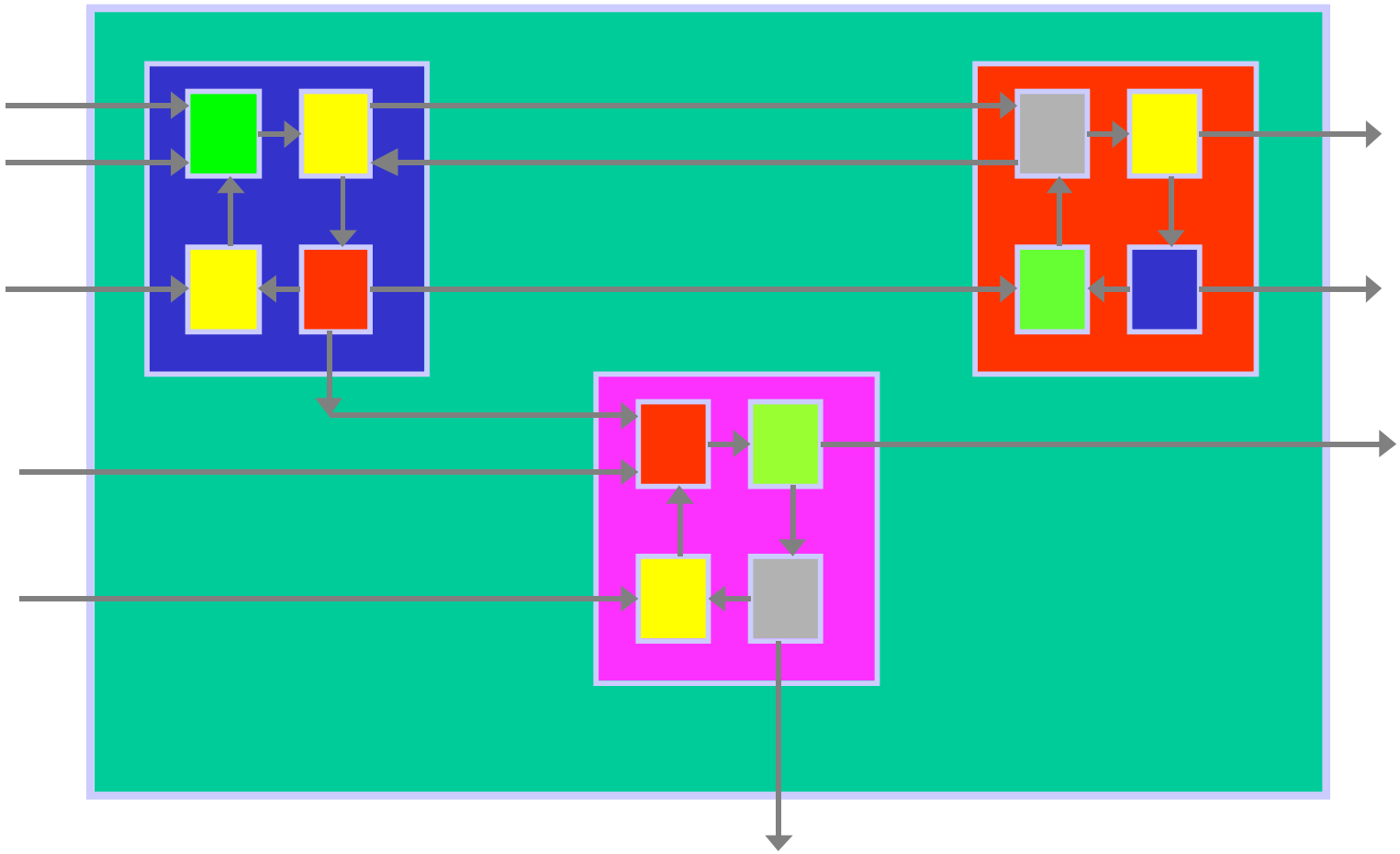
$$\{m\}\#\overline{r\downarrow t} = 0 \Rightarrow \{m\}\#\overline{x\downarrow t+1} = 0$$

that allows us to conclude

$$\bar{y} = \langle\rangle$$

which cannot be concluded without strong causality.

Local Timing



Hierarchical Architectures of Components with Different Time Scales

Local Timing

- For each subsystem (subcomponent) of a network we can introduce a local time and a local time granularity.
- **Example**: Architecture with local Time Granularities
- The following expression

$$COA(COA(F \otimes G, 10) \otimes COA(H \otimes K, 20) \otimes L, 2)$$

- denotes a system with a subsystem
  - $F \otimes G$ which works in a 10 times faster mode
  - $H \otimes K$ which works in a 20 times faster mode
- This way we get a system with different local time scales where
  - each time scale can be chosen fine enough for each of the components locally
  - to guarantee strong causality of each of its subcomponents such that all local computations are captured on the right level of time granularity.

Conclusion: Robust Flexible Timing

- Relationship between causality and the choice of the time scale and granularity as well as its influence onto compositionality.
- If we choose the time scale fine enough the definition of faithful composition is quite straightforward.
- Our approach supports flexibly chosen time scales. We worked out the following idea of flexible timing:
  - The leaves of the component hierarchy are state machines.
  - Each state machine runs in its own local time scale that defines the time duration of each of its steps
  - Each processor (CPU, controller) contains a family of state machines which run with different speed (time granularity). This defines the finest time grain steps and the set of steps (state transitions) that have to be executed (scheduled) in a time slot. This determines the workload.
- We can design a static schedule at the abstract level of behavior without being forced to address low-level technical issues such as schedulers or operating systems.

Conclusion: Robust Flexible Timing

- We obtain a flexible and modular theory of timing of systems and sub-systems this way which provides the following helpful properties
  - high level abstract and flexible modeling
  - multiplexing and scheduling on abstract level
  - application oriented time model close to hardware time model
  - causality for inductive reasoning
  - avoiding causal loops
  - rich algebraic properties
  - discrete model of time that provides the same flexibility as analog models of time (real time)
  - time abstraction by coarsening the time granularity to get rid of unwanted delays.
- In the end we can generate from such a time model a static scheduling for the system.