Notes on delay-insensitive communication


Document status and date:
Published: 01/01/1988

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Notes on Delay-Insensitive Communication
by
Huub M.J.L. Schols
88/06

March 1988
 This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science of Eindhoven University of Technology. Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review. Copies of these notes are available from the author or the editor.

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB Eindhoven
The Netherlands
All rights reserved
editor: F.A.J. van Neerven
Notes on Delay-Insensitive Communication

Huub M.J.L. Schols
Institute for Biomedical Computing
Washington University
St.Louis, Missouri, U.S.A.

and
Department of Mathematics and Computing Science
Eindhoven University of Technology
Eindhoven, the Netherlands

0 Introduction

In this monograph we address the communication between mechanisms. Mechanisms communicate by sending and receiving (physical) signals. Our notion of delay-insensitive communication comprises that the value of the delay between the sending and the reception of each such signal is unknown.

We introduce a formal communication model. In the communication model we use trace theory, a formalism developed at Eindhoven University of Technology by Martin Rem e.a., cf. [Rem, Rem-van de Snepscheut-Udding, van de Snepscheut, Kaldewaij], as a tool. The interpretation of trace theory in the communication model yields a formalization of delay-insensitive communication. As a consequence of this approach, our research is concerned with three topics:

(#) Part of this work was supported by the U.S. Public Health Service under Grant No. R24 RR01379.
the trace theory formalism,
the relation between the communication model and the underlying physics, and
the interpretation of the trace theory formalism in the communication model.

In section 1 we present some tools that we use. The communication model is presented in section 2. In section 3 we are concerned with delay-insensitive communication. Furthermore, we address transmission hazard; we say that there exists no transmission hazard, if and only if no two signals are able to interfere with each other. In section 4 we focus our attention on the communication behavior of mechanisms that communicate in a delay-insensitive way. We take absence of computation hazard into account; by absence of computation hazard we mean that no signal can arrive at a moment that it cannot be accepted. We consider absence of unspecified termination hazard in section 5; unspecified termination hazard is a liveness criterion. It is related to deadlock and livelock.

There may exist various reasons why one is interested in delay-insensitive communication, cf. [Barney]. Here, we mention scaling, variable or unknown delays, and metastability. When integrating circuits at a very large scale, delays that occur in the interconnections between the switching elements tend to increase relatively to the delays in the switching elements, cf. [Seitz80, van de Snepscheut]. Dealing with unknown delays enables us to separate the functional and the topological design tasks; in particular, it yields a lot of freedom for placement and routing. Chaney and Molnar demonstrated, cf. [Chaney-Molnar], that metastable behavior is an important and intrinsic issue; therefore we mention it next to (other) variable delays; see also [Kleemann-Cantoni]. Another field for application of delay-insensitive communication is the communication in distributed systems, e.g. transputers, n-cube, or the FFP-machine.

0.0 Notions related to delay-insensitivity

Many people have been concerned with concepts that are related to delay-insensitivity. In the literature one encounters a variety of names: asynchronous, speed-independent, self-timed, delay-insensitive. Although distinct names are used, people are dealing with related intuitive notions. Attempts have been made to formalize these notions stressing distinct characteristics. Furthermore, the same name might be used by distinct people to indicate different aspects of the intuitive notions. The noun asynchronous arose to distinguish between synchronous, e.g. globally clocked, and not synchronous, e.g. locally clocked or not clocked, systems, cf. [Muller-Sartky, Unger, Rosenberger, Molnar-Fang, Keller75, Dill-Clarke, Molnar]. Muller, cf. [Miller], Keller, cf. [Keller74], Fang, and Molnar, cf. [Fang-Molnar], use speed-independent, and Seitz is among others concerned with self-timed systems, cf. [Seitz79, Martin85, Yakovlev, Greenstreet-Williams-Staunstrup]. Van de Snepscheut and Martin both use delay-insensitive. They stress the internal communication, cf. [van de Snepscheut, Martin86]. Since they deal with so-
called open systems, the external communication between the mechanisms and an external environment need not be delay-insensitive. Molnar, Fang, and Rosenberger apply delay-insensitivity to the external communication of their Macromodules, cf. [Molnar-Fang-Rosenberger, Molnar-Fang]. The internal communication is, generally, not delay-insensitive. Based upon this approach several formalizations have emerged, cf. [Udding, Schols, Verhoeuff, Ebergen]. Although the reasoning in formalizing their intuitive notions is very different from each other, they come up with equivalent formalizations.

1 Tools and denotations

In this section we present some tools that we use in the remainder of this monograph. In subsection 1.0 we explain some linguistic denotations.

1.0 Denotations in the English language

We use double quotes to indicate that we refer to the enclosed passage as a noun, not as a part of the sentence. Single quotes are used to indicate that we are skeptical about the enclosed passage. We use underlining to stress a part of a sentence. Italics are used to indicate the first appearance and/or definition of the formal notions in this monograph.

We also use italics and boldface printing to distinguish the formal objects from the words in the English language.

1.1 Set theory

We use pairs of curly brackets to denote sets. The elements of a set are listed between the curly brackets and are separated from each other by commas. We denote the empty set by \( \emptyset \). The binary relation "is an element of" is denoted by the infix operator \( \in \).

In this monograph all variables that range over numbers, range over the natural numbers.

example

\( \{3, 8\} \) denotes the set that consists of the natural numbers 3 and 8.

end of example

1.2 Quantification

Universal quantification, i.e. generalized conjunction, is denoted by \( (A l: R: E) \), where \( A \) is the quantifier, \( l \) is the list of bound variables, \( R \) is the range of the variables, and \( E \) is the quantified expression. Analogously, we denote existential quantification, i.e. generalized disjunction, by \( (El: R: E) \). We also use quantification to denote sets: \( \{l: R: e\} \), where \( e \) denotes an element of the set.
Notes on Delay-Insensitive Communication

example

(Ai : 6 ≤ i < 9 : Pi) denotes P_6 \land P_7 \land P_8.
(Ei , j : (2 ≤ i ≤ 5) \land EVEN(i) \land (i = j) : S_i) denotes S_2 \lor S_4.
{i : 2 ≤ i ≤ 4 : i^2} denotes \{4, 9, 16\}.

end of example

1.3 Trace theory

We present some formal notions that occur in trace theory. Trace theory is a tool that has been used to formalize communication. We use trace theory to formalize delay-insensitive communication. For an extended overview of trace theory we refer to [Rem, Rem-van de Snepscheut-Udding, van de Snepscheut, Kaldewaij].

remark

Mazurkiewicz has developed a formalism that is also called trace theory, cf. [Mazurkiewicz]. Mazurkiewicz's trace theory differs from our trace theory. In some sense Mazurkiewicz's traces can be interpreted as equivalence classes of our traces.

end of remark

We use finite trace theory, i.e. all traces have finite length. Van Horn, cf. [Van Horn], and Black, cf. [Black], extend finite trace theory with infinite traces in order to deal with liveness properties. In section 5 we show that it is possible to discuss some liveness properties within finite trace theory.

1.3.0 Basic notions of trace theory

We assume that there exists a universe \( \Omega \) of symbols, that is large enough, i.e. we will not run out of symbols. An alphabet is a finite subset of \( \Omega \). We denote the sequence of zero symbols by \( e \). Catenation is denoted by juxtaposition. The Kleene-closure of a set of symbols is the set of all finite-length sequences of symbols.

definition

For a set \( B \) of symbols the Kleene-closure of \( B \), denoted by \( B^* \), is defined recursively by:

(i) \( e \in B^* \),

(ii) \( (As , a : s \in B^* \land a \in B : sa \in B^*) \),

(iii) completeness axiom: \( B^* \) contains no elements that are not required by (i) or (ii).

end of definition
The elements of $\Omega^*$ are called traces. We refer to $\varepsilon$ as the empty trace. A set of traces is called a trace set.

### 1.3.1 Directed trace structures

A directed trace structure is a triple $<A , T , D >$. $A$ denotes the alphabet of the directed trace structure. $T$ denotes the trace set of the directed trace structure; it is a subset of $A^*$. $D$ denotes the direction of the directed trace structure; it is a partition of $A$; in this monograph we restrict the directions to bipartitions, i.e. partitions into two disjunct parts. Hence, the intersection of the two parts is empty and their union is $A$. For a directed trace structure $S$, $aS$ denotes the alphabet of $S$, $tS$ denotes the trace set of $S$, and $dS$ denotes the direction of $S$. We say that symbols $a$ and $b$ have the same type w.r.t. a direction $D$, if they are in the same part of the partition indicated by $D$. Otherwise, we say that $a$ and $b$ have distinct types w.r.t. $D$, or $a$ has another type than $b$ w.r.t. $D$.

**remark**

The definition of directed trace structure differs from the definitions that have been used in the past, cf. [Udding, Schols, Verhoeff, Ebergen]. There are two differences:
- an interpretational issue: we introduce a separation of concerns between the formal object, i.e. the directed trace structure, and its interpretation; we relate the notions input and output to the interpretation of a directed trace structure, not to a directed trace structure itself!
- a denotational issue: this is caused by the first difference.

Since there exist mappings from the former definition into our communication model, in which the directed trace structures defined above are used, we did not loose any expressive power.

**end of remark**

**example**

For symbols $a$ and $b$, and directed trace structure $S$, such that

$$S = < \{a,b\}, \{\varepsilon, a, b, ab\}, \{a, b\} \oplus \emptyset >,$$

the symbols $a$ and $b$ have the same type w.r.t. $dS$.

**end of example**

In the example above we show the usage of the symmetric operator $\oplus$ to construct a direction from two disjunct alphabets.

### 1.3.2 Some notions in trace theory

For trace $t$ and symbol $a$ we denote the number of occurrences of $a$ in $t$ by $\#_at$. 
definition
We define the number of occurrences of a symbol in a trace recursively by:

(i) for symbol $a$, 
$$\#_a \overset{\text{def}}{=} 0,$$
(ii) for trace $t$ and symbol $a$, 
$$\#_a(ta) \overset{\text{def}}{=} (\#_a t) + 1,$$
(iii) for trace $t$ and distinct symbols $a$ and $b$, 
$$\#_a(tb) \overset{\text{def}}{=} \#_a t.$$
end of definition

For traces we define their bag:

definition
For trace $t$, bag $t$ denotes the bag of $t$:
$$\text{bag } t \overset{\text{def}}{=} \{a : a \in \Omega : (a, \#_a t)\}.$$
end of definition

We define the binary operation prefix on traces.

definition
For traces $s$ and $t$, $s$ is called a prefix of $t$, denoted by $s \text{ prefix } t$, if and only if 
$$(\exists u : u \in \Omega^* : su = t).$$
end of definition

For trace sets we define the unary operation prefix-closure.

definition
For trace set $T$, pref$T$ denotes the set of all prefixes of $T$:
$$\text{pref } T \overset{\text{def}}{=} \{s, t : t \in T \land \text{prefix } t : s\}.$$
end of definition

We call a trace set, say $T$, prefix-closed, if and only if $T = \text{pref } T$. We extend the definition of prefix-closure to directed trace structures.

definition
For directed trace structure $S$, pref$S$ denotes the prefix-closure of $S$:
$$\text{pref } S \overset{\text{def}}{=} < aS, \text{pref } (tS), dS >.$$
end of definition

We call a directed trace structure, say $S$, prefix-closed, if and only if $S = \text{pref } S$. We call a directed trace structure nonempty, if and only if its trace set is nonempty.
We extend the definition of intersection to directed trace structures.

**Definition**

For directed trace structures $S$ and $T$, such that $aS = aT$ and $dS = dT$, the intersection of $S$ and $T$, denoted with $S \cap T$, is defined by

$$S \cap T \overset{\text{def}}{=} <aS, tS \cap tT, dT>.$$  

**end of definition**

Analogously to subsets we define the notion sub-directed-trace-structures.

**Definition**

For directed trace structures $S$ and $T$, such that $aS = aT$ and $dS = dT$, we call $S$ a sub-directed-trace-structure of $T$, denoted with $S \subseteq T$, if and only if $tS \subseteq tT$.

**end of definition**

We introduce the notion i/o-direction. An i/o-direction is a pair of disjunct sets of symbols, which are called the input alphabet and the output alphabet of the i/o-direction. As such, an i/o-direction is a bipartition of the union of these sets. Given i/o-direction $F$, the input alphabet of $F$ is denoted by $iF$ and the output alphabet of $F$ is denoted by $oF$.

We define the reflection of an i/o-direction:

**Definition**

For an i/o-direction $F$, the reflection of $F$, which is denoted by $F$, is defined by

$$iF \overset{\text{def}}{=} oF$$

$$oF \overset{\text{def}}{=} iF$$

**end of definition**

Notice that the reflection of an i/o-direction is an i/o-direction.

### 1.3.3 Notational convention

Small letters near the beginning of the Latin alphabet are symbols; when they are used as variables, they denote symbols. Small letters near the end of this alphabet denote traces. Capital letters are used to denote alphabets, directions, i/o-directions, and directed trace structures.

### 1.3.4 Trace theory properties

We often refer to directed trace structures that contain $e$ in their trace set and that are prefix-closed.
property

For a prefix-closed directed trace structure $S$,

$$(e \in tS) = (S \text{ is nonempty}).$$

end of property

On account of the property above we may refer to such directed trace structures as directed trace structures that are nonempty and prefix-closed.

2 The communication model

We introduce the communication model to achieve a separation of concerns between the interpretation of the underlying physics and the usage of the trace theory formalism. When one addresses communication in a formal way, one introduces an abstraction from the underlying physics; the latter is either some physical model, that is considered to constitute a good model for some physical phenomena, or it is one’s notion of ‘physical reality’. We want to abstract from such underlying physics, but we do not interpret any notions of trace theory in the underlying physics directly: we interpret them in the communication model.

Van de Snepscheut, cf. [Van de Snepscheut], and Udding, cf. [Udding], try to establish such a separation of concerns by carefully separating their trace theory formalism from its mechanistic appreciation. By the introduction of the communication model this separation of concerns has become explicit.

2.0 Communication

By physical communication we mean the exchange of signals between two or more mechanisms. We restrict ourselves to physical communication between two mechanisms. In the communication model we call a mechanism a component. Components are connected to each other by formal paths, see figure 0. The connection between a component and a path is called a port. An event is a possible communication action. With each port of a component we associate a distinct event. Mechanisms communicating signals (physically) are modeled by components communicating instances of events at their ports (formally) in the communication model. Events are generic communication actions; instances of events are particular occurrences of them. Notice that the communication model is an event-based model: the instances of events are atomic.

remark

Restricting ourselves to considering communication between two mechanisms, does not exclude broadcasting or buswire communication from the description power of the communication model. We deal with these communication forms by
introducing components for them. These are connected to other components by paths. E.g. the Buswire described by Molnar, cf. [Molnar], will be dealt with in [Ebergen-Molnar-Schols].

end of remark

The set of paths that connect two components is called the \textit{channel} (between those two components).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure0.png}
\caption{Components $\Gamma$ and $\Delta$ and the channel between them.}
\end{figure}

In figure 0 we present a schematic of two components and the channel between them. A \textit{communication signal} is a formal notion in the communication model; it is said to travel along a path from the port at which it is sent to the port at which it is received. Communication signals propagate monotonically: when at some moment a communication signal has propagated to a certain point of a path, at each later moment it will have propagated further and, eventually, it will reach the port at which it will be received. The paths are unidirectional, i.e. all communication signals that propagate along the same path propagate in the same direction. With each path of a channel we associate a distinct event; at any path one point is chosen; all instances of events are said to occur at the point that is chosen at the path with whom the event is associated. In [Schols] we showed that the formalization of delay-insensitive communication in a channel does not depend on a particular choice of these points; the only requirement is that exactly one point is chosen at each path of the channel. The communication between two mechanisms is modeled by instances of events occurring in the \textit{paths} of the channel between the corresponding components.
### Modeling communication

<table>
<thead>
<tr>
<th>'physics'</th>
<th>communication model</th>
</tr>
</thead>
<tbody>
<tr>
<td>mechanism</td>
<td>component</td>
</tr>
<tr>
<td>mechanisms communicate</td>
<td>components communicate via paths and/or</td>
</tr>
<tr>
<td></td>
<td>at coinciding boundaries</td>
</tr>
<tr>
<td>port</td>
<td></td>
</tr>
<tr>
<td>event</td>
<td></td>
</tr>
<tr>
<td>instance of an event</td>
<td></td>
</tr>
<tr>
<td>mechanism communicates signals</td>
<td>component communicates instances of events</td>
</tr>
<tr>
<td></td>
<td>communication signal</td>
</tr>
<tr>
<td></td>
<td>communication signals propagate monotonically</td>
</tr>
</tbody>
</table>

**Table 0**  
Relation between communication model and underlying physics.

In table 0 we present a schematic of the relation between the communication model and the underlying physics. The blank entries in the left column show a dependency on the particular physical model or 'physical reality' that one considers as the underlying physics.

**Remark**

We discuss "communication behavior of components" and "communication between components". We do not discuss "observation" nor problems related to observing communication. Although it is possible to discuss some observation issues within the communication model, cf. [ScholsIPb], we will not do so in this monograph.

**End of Remark**

Signals that occur in parallel or concurrently are modeled to occur independently, i.e. there exists no causal relation between them. For a more extended treatment hereof we refer to [ScholsIPa].

When we need variables in the communication model we use Greek letters that do not occur in the Latin alphabet. We do not use ε or Ω, since they are used for other purposes in the trace theory formalism, see subsection 1.3.0.

### 2.1 Interpretation of the basic notions of trace theory

Symbols are used to denote events or instances of events. An event and all its instances are denoted by the same symbol! A trace denotes a sequence of instances of events; every symbol in a trace denotes a distinct instance of the event denoted by that symbol. The occurrence in a communication of an instance α before an instance β is modeled by:
the symbol that denotes the particular instance $\alpha$ occurs in a trace to the left of the symbol that denotes the particular instance $\beta$.

dexample

Symbols $a$ and $b$ denote instances $\alpha$ and $\beta$, respectively. The occurrence of $\alpha$ before $\beta$ is modeled in the trace $ab$.

deend of example

dremark

Notice, that the relationship between the ordering of instances in the communication model and the relative positions of symbols in a trace in the trace theory formalism is one-sided. The occurrence in a trace of a symbol that denotes an instance $\alpha$ to the left of a symbol that denotes an instance $\beta$, leads to the interpretation that $\beta$ does not occur before $\alpha$ in the communication that is modeled by the trace. Instance $\alpha$ need not occur before instance $\beta$ in this communication.

deend of remark

2.2 Interpretation of directed trace structures

We shall use directed trace structures to model communication. We shall associate a directed trace structure either with a channel or with a component, see subsections 2.2.0 and 2.2.1 respectively. In subsection 2.2.2 we deal with some interpretational issues that apply in both cases: when the directed trace structure is associated with a channel and when the directed trace structure is associated with a component.

2.2.0 Directed trace structures associated with a channel

When we associate a directed trace structure with a channel, the directed trace structure models the communication along the paths of the channel. The interpretation of the trace set of the directed trace structure is: if a trace of symbols is not in the trace set, then the corresponding sequence of instances of events does not occur in the channel. Given channel $\Gamma$, $\text{ptr}\Gamma$ denotes the directed trace structure that is associated with $\Gamma$. The paths in a channel are unidirectional; as a consequence there are two directions, which are opposite to each other: when we wish to distinguish between them we call them back and forth. The set of symbols, that denote the events that are associated with the paths in the back direction of a channel $\Gamma$, is called the back alphabet of $\Gamma$, which is denoted by $b\Gamma$; the set of symbols, that denote the events that are associated with the paths in the forth direction of a channel $\Gamma$, is called the forth alphabet of $\Gamma$, which is denoted by $f\Gamma$. Of course, $d(\text{ptr}\Gamma)=(b\Gamma \oplus f\Gamma)$ and $a(\text{ptr}\Gamma)=(b\Gamma \cup f\Gamma)$.

Notice that back and forth are merely nouns used to distinguish between the two directions: they can only be used to indicate equal or opposite directions. Notice the difference with input and output, see subsection 2.2.1.
2.2.1 Directed trace structures associated with a component

When we associate a directed trace structure with a component, the directed trace structure models the communication behavior of the component at the ports of the component. Its alphabet is interpreted as a denotation of the set of events in which the component is allowed to engage. Since components can only communicate via unidirectional paths, we distinguish two types of events in which a component is allowed to engage: input and output. The set of symbols, that denote the events that can be received by a component $\Gamma$, is called the input alphabet of $\Gamma$, which is denoted by $i\Gamma$; the set of symbols, that denote the events that can be sent by a component $\Gamma$, is called the output alphabet of $\Gamma$, which is denoted by $o\Gamma$. Given component $\Gamma$, $\ptr \Gamma$ denotes the directed trace structure that is associated with $\Gamma$. Of course, $d(\ptr \Gamma)=(i\Gamma \oplus o\Gamma)$ and $a(\ptr \Gamma)=(i\Gamma \cup o\Gamma)$.

Notice that input and output alphabets are, due to their interpretation, uninterchangeable.

The interpretation of the trace set of the directed trace structure is: the component does not engage in a sequence of instances of events if the corresponding trace of symbols is not in the trace set. Notice that this interpretation of the trace set imposes the obligation to the components to accept inputs; it does not impose any obligation to the components to send outputs!

We call an i/o-direction, say $F$, the i/o-direction of $\Gamma$, if and only if $iF=i\Gamma$ and $oF=o\Gamma$.

2.2.2 General interpretational issues

We consider directed trace structures that are associated with a channel or a component. In order to model that initially no instance of any event has occurred we require that the trace set of such a directed trace structure contains the empty trace. Furthermore, if instance $\alpha$ occurs before $\beta$, it is possible that $\alpha$ has occurred and $\beta$ has not (yet) occurred. As a consequence, we require that the trace set of such a directed trace structure is prefix-closed.

property

For $\Gamma$ a channel or a component,
\[ \ptr \Gamma \] is prefix-closed and nonempty.

end of property

Traces are linear objects. In order to model that instances $\alpha$ and $\beta$ occur independently, we include in the trace set as well traces in which the symbol that denotes $\alpha$ occurs to the left of the symbol that denotes $\beta$ as traces in which the symbol that denotes $\beta$ occurs to the left of the symbol that denotes $\alpha$. 


example

A component has two events that are denoted by the symbols $a$ and $b$. Symbols $a$ and $b$ have the same type. Only one instance of each event can occur. They occur independently of each other. The directed trace structure that denotes the communication behavior of this component is:

$$<\{a,b\}, \{e,a,b,ab,ba\}, \{a,b\} \oplus \emptyset>.$$  

end of example

The interpretation of directed trace structures in the communication model is illustrated by the following example.

denoted by the following example.

declared by the following example.

example

Given the directed trace structure

$$<\{a,b,c\}, \{e,a,b,ab,ba,abc,bac\}, \{a,b\} \oplus \{c\}>,$$

where $a$, $b$, and $c$ denote the events $\alpha$, $\beta$, and $\gamma$ respectively, and their respective instances. In the communication model we interpret this directed trace structure as follows:

- there are three events, viz. $\alpha$, $\beta$, and $\gamma$;
- at most one instance of each event can occur;
- the instances of the events $\alpha$ and $\beta$ are not ordered in the communication; they both occur before the instance of the event $\gamma$ occurs.
- if this directed trace structure is associated with a channel, then the paths, with which $\alpha$ and $\beta$ are associated, have a direction opposite to the path, with which $\gamma$ is associated;
- if this directed trace structure is associated with a component, then either $\alpha$ and $\beta$ are inputs and $\gamma$ is an output or $\alpha$ and $\beta$ are outputs and $\gamma$ is an input.

Which of the two alternatives holds is an interpretational issue; this is not reflected by the directed trace structure.

end of example

2.3 Denotation of events

Although we use distinct symbols in the trace theory formalism to denote the events of a component or a channel, the same symbol is often used to denote an event of a component as well as an event of another component as well as an event of a channel. We illustrate this by the following example.

denoted by the following example.

example

We consider two components $\Gamma$ and $\Delta$ connected by channel $\Theta$, see figure 1.
Components $\Gamma$ and $\Delta$ and channel $\Theta$ between them.

$\Gamma$ and $\Delta$ have three ports each. Events $\alpha$, $\beta$, and $\gamma$ are associated with the three ports of $\Gamma$. The events $\delta$, $\zeta$, and $\eta$ are associated with the three ports of $\Delta$. The events $\theta$, $\iota$, and $\kappa$ are associated with the three paths of channel $\Theta$. The path with event $\theta$ connects the output port of $\Gamma$ with event $\alpha$ to the input port of $\Delta$ with event $\delta$; the path with event $\iota$ connects the output port of $\Gamma$ with event $\beta$ to the input port of $\Delta$ with event $\zeta$; the path with event $\kappa$ connects the output port of $\Delta$ with event $\eta$ to the input port of $\Gamma$ with event $\gamma$.

In the trace theory formalism, we use symbol $a$ to denote $\alpha$, $\theta$, and $\delta$; symbol $b$ denotes $\beta$, $\iota$, and $\zeta$ and symbol $c$ denotes $\eta$, $\kappa$, and $\gamma$. Notice that all events of component $\Gamma$ are denoted by distinct symbols, as are all events of component $\Delta$ and all events of channel $\Theta$.

We consider directed trace structure $S$:

$$S = \langle \{a, b, c\}, \{e, a, b, ac, bc\}, \{a, b\} \oplus \{c\} \rangle.$$  

When we associate $S$ with component $\Gamma$, the interpretation of $S$ is: $\Gamma$ may send an instance of $\alpha$ or an instance of $\beta$ (not both!), after which $\Gamma$ is able to accept an instance of $\gamma$. When we associate $S$ with component $\Delta$, we find the analogous interpretation of $S$: $\Delta$ is able to accept an instance of $\delta$ or an instance of $\zeta$ (not both!), after which $\Delta$ may send an instance of $\eta$. When we associate $S$ with channel $\Theta$, we find the interpretation of $S$: either an instance of $\theta$ or an instance of $\iota$ may occur (not both!) after which an instance of $\kappa$ may occur.

end of example
2.4 Hazards

Given a hazard, e.g. computation interference, we refer to "absence of computation hazard" rather than to "absence of computation interference". The reason is: we do not only want to exclude computation interference, but we also want to guarantee that no computation interference can occur. The same thing applies to "absence of transmission hazard" and "absence c.: unspecified termination hazard". The name hazard originates from switching theory; there it was introduced with the same connotation that we attach to it.

example

We consider components $\Gamma$ and $\Delta$. Each has one input port and one output port.

We consider instances $\alpha$, $\beta$, $\gamma$, and $\delta$. $\alpha$ can be sent by $\Gamma$; $\beta$ can be received by $\Gamma$; $\gamma$ can be received by $\Delta$; $\delta$ can be sent by $\Delta$. There exists a path from the port, to which $\alpha$ is related, to the port to which $\gamma$ is related, and there exists a path from the port, to which $\delta$ is related, to the port to which $\beta$ is related, see figure 2. Symbol $a$ denotes the instances $\alpha$ and $\gamma$; symbol $b$ denotes the instances $\beta$ and $\delta$. The communication behaviors of $\Gamma$ and $\Delta$ are given by:

- $\text{of} \defeq \{ a \}$,
- $\text{i} \Gamma \defeq \{ b \}$,
- $\text{i} \Delta \defeq \{ a \}$,
- $\text{o} \Delta \defeq \{ b \}$,
- $\text{ptr} \Gamma \defeq \langle [a, b], [e, a, ab], \{ a \} \oplus \{ b \} \rangle$,
- $\text{ptr} \Delta \defeq \langle [a, b], [e, a, b, ab, ba], \{ a \} \oplus \{ b \} \rangle$.
Initially, $\Delta$ may send $\delta$; although $\Gamma$ is able to accept $\beta$ after $\Gamma$ has sent $\alpha$, it is possible that $\beta$ is received by $\Gamma$ before $\Gamma$ has sent $\alpha$. This situation we call "computation hazard".

end of example

3 Formalizations of delay-insensitive communication

The formalizations of delay-insensitive communication are based, cf. [Udding, Schols, Verhoeff, Ebergen], on the Foam Rubber Wrapper metaphor, which was introduced by Charles Molnar, cf. [Molnar-Fang-Rosenberger].

3.0 The Foam Rubber Wrapper metaphor

We call the communication in a channel delay-insensitive if the one and only assumption made w.r.t. the delays of the communication signals that travel along the paths of the channel is: these delays are nonnegative. This assumption reflects that no communication signal is received before it has been sent. Even distinct instances of the same event, which travel along the same path, may have different values of delays.

Our notion of delay-insensitive communication has emerged from the Foam Rubber Wrapper metaphor. According to this metaphor, a wrapper of foam rubber enwraps a component. The drawing in figure 3 has been made such that we cannot tell which component is wrapped, cf. [Molnar-Fang-Rosenberger].
We observe that actually the paths of the channel are inside the Foam Rubber Wrapper, see figure 3. This observation leads to the characterization of delay-insensitive communication at the beginning of this section. Notice that delay-insensitivity is not a property of (the communication behavior of) a component, but it is a property of the communication (in the channel) between components.

3.1 Causality

Throughout this subsection we consider the interconnected components $\Gamma$ and $\Delta$, see figure 4.

Components $\Gamma$ and $\Delta$ are connected, such that $\Omega = \alpha \Delta$ and $\omega = \iota \Delta$. As a consequence, $\alpha \Gamma = \alpha \Delta$. Let $F$ be the i/o-direction of $\alpha \Gamma$.

We have stated in subsection 3.0 that the delays of the communication signals in the paths are nonnegative. We model this causality by defining the composability relation between traces. We call a trace $t$ composable under $F$ with a trace $u$ if, at some moment, $t$ models the order of instances of events at the ports of $\Gamma$ and $u$ models the order of instances of events at the ports of $\Delta$.

**definition**

Given i/o-direction $F$ and traces $t$ and $u$, we call $t$ composable under $F$ with $u$, denoted by $t \in F u$, if and only if

\[(\forall a : a \in \alpha F : \#_a t \geq \#_a u)\]

\[\land (\forall b : b \in \iota F : \#_b u \geq \#_b t)\]
Notes on Delay-Insensitive Communication

\[ (a, b, r, s : a \in oF \land b \in iF \land rbprefix \land asprefix u : (\#_a r > \#_a s) \lor (\#_b s > \#_b r)) \]

end of definition

In the definition above, the first two conjuncts reflect that all received instances of events have been sent. The third conjunct reflects that when two communication signals travel in opposite directions, at least one of them has to be sent before either of them is received.

In order to illustrate the third conjunct, we consider instances \( \alpha, \beta, \gamma, \) and \( \delta \). \( \alpha \) can be sent by \( \Gamma \); \( \beta \) can be received by \( \Gamma \); \( \gamma \) can be received by \( \Delta \); \( \delta \) can be sent by \( \Delta \). There exists a path from the port, to which \( \alpha \) is related, to the port to which \( \gamma \) is related, and there exists a path from the port, to which \( \delta \) is related, to the port to which \( \beta \) is related, see figure 5. Now the third conjunct in the definition of composability reflects that in this communication it is excluded that \( \alpha \) does not occur before \( \beta \) and \( \delta \) does not occur before \( \gamma \).

example

For i/o-direction \( F \), such that \( oF = \{a\} \) and \( iF = \{b\} \),

\[ abcFba \land \neg (abcFba) \]

end of example

From the example above we infer that, in general, the definition of composable traces is not symmetric in inputs and outputs.
property

For traces $t$ and $u$ and i/o-direction $F$,

$$tcFu = ucpt.$$ 

end of property

However, the property above reflects that there exists some symmetry w.r.t. inputs and outputs.

remark

Notice that the definition of composability is not restricted to traces of given trace sets. The i/o-direction is important. Although it is possible to restrict the definition to traces of given trace sets, cf. [Udding, Schols], the approach we show in this monograph leads to a mathematically nicer formalism, cf. [Verhoeff].

end of remark

3.2 Discussion of formal definitions of delay-insensitivity

Based upon the Foam Rubber Wrapper metaphor four formalizations of delay-insensitivity arose. Udding, the first one to capture delay-insensitivity formally, has presented a set of rules, i.e. predicates on directed trace structures, that are necessary and sufficient for delay-insensitivity, cf. [Udding]. We have defined a composition operator that can be used to check delay-insensitivity and also to compute the smallest delay-insensitive communication that includes the original communication, cf. [Schols]. Ebergen has defined Wire components, cf. [Ebergen]. He does not distinguish the paths of the channels from the other components; in his approach transmission hazard is a special case of computation hazard. Verhoeff has introduced a Delay-demon, that models the delays in the paths of the channel, cf. [Verhoeff]. All four formalizations turn out to be equivalent.

Although the definitions of the four formalizations differ very much, they are all inconvenient to prove by hand that a particular communication is delay-insensitive. All can be used to show that a particular communication is not delay-insensitive: find a case and show that it is an exception. Burstyn and Udding have written a program, to test automatically if the composition of a number of delay-insensitive modules is correct in the sense that no possible sequence of communications can result in computation hazard. This program can be used to verify that a particular communication is delay-insensitive, cf. [Burstyn].

In [Schols] the class of all directed trace structures that satisfy the Foam Rubber Wrapper principle is defined. We call this class $D_4$, cf. [Schols, Siccama, Verhoeff].
Notes on Delay-Insensitive Communication

3.3 Analysis of delay-insensitive communication

We shall carefully distinguish between "communication in a channel" and "communication behavior of a component". These two topics are, of course, related to each other. Distinguishing these two topics enables us to separate the communication behavior of components from the delay requirements in the channel; furthermore, it enables us to discuss a liveness criterion, viz. absence of unspecified termination hazard, see section 5.

3.3.0 Delay-insensitivity

As analyzed in subsection 3.0 delay-insensitivity is characterized by the values of the delays of the communication signals traveling along the paths of a channel. As a consequence, delay-insensitivity is a property of the communication in a channel; it is not a property of the communication behavior of a component.

We call the communication in a channel delay-insensitive, if and only if the directed trace structure that denotes this communication is an element of $D_4$.

We use the name delay-insensitive channel as an abbreviation for "channel in which the communication is delay-insensitive".

3.3.1 Impact of delay-insensitive communication on components

We consider a component $\Gamma$ that is connected to a channel, see figure 6.
We assume that the communication in the channel is delay-insensitive. This reduces the actual communication behavior of \( r \). E.g. if \( r \) is connected to a delay-insensitive channel, then in the actual communication behavior of \( r \):
- no input enables an input,
- no output disables an input.

For a proof of these properties we refer to [ScholsIPa].

3.3.2 Absence of transmission hazard

We characterized absence of transmission hazard by: no two signals can interfere with each other. Within the realm of delay-insensitive communication absence of transmission hazard is equal to: no communication signal is sent along a path before all communication signals, that previously have been sent along that path, have been received. The subclass of \( D_4 \) that models absence of transmission hazard is called \( C_4 \), cf. [Udding].

definition
A directed trace structure \( T \) is an element of \( C_4 \), if and only if
\[
T \in D_4 \land (A s, a : s \in (aT)^* \land a \in aT : saa \notin tT).
\]
end of definition
definition

The delay-insensitive communication in a channel has *absence of transmission hazard*, if and only if the directed trace structure that denotes this communication is an element of \( C_4 \).

end of definition

We deal with absence of transmission hazard as a property that may or may not hold for delay-insensitive communication. However, Udding, cf. [Udding], and Ebergen, cf. [Ebergen], include absence of transmission hazard in their formalizations of delay-insensitivity.

4 Communication behavior of a component

As discussed in section 2, we associate a directed trace structure with a component to model the communication behavior of the component. In section 3 we showed that delay-insensitivity is a property of the communication in a channel between components; furthermore, we argued that there exists an impact of delay-insensitive communication on the communication behavior of a component. In this section we focus on the communication behavior of a component that communicates delay-insensitively.

4.0 Computation hazard

In this monograph we define *absence of computation hazard* for components that communicate delay-insensitively. For a general definition of absence of computation hazard we refer to [ScholsIPa].

definition

Given components \( \Gamma \) and \( \Delta \) and i/o-direction \( F \), such that \( F \) is the i/o-direction of \( \Gamma \) and \( \bar{F} \) is the i/o-direction of \( \Delta \). \( \Gamma \) and \( \Delta \) have no computation hazard, if and only if

\[
( A \tau, u, a : t \in t(ptr\Gamma) \land t_F = u \land t(ptr\Delta) \land a \in F \land \#_a t < \#_a u : ta \in t(ptr\Gamma))
\]

\[
\land ( A \tau, u, b : t \in t(ptr\Gamma) \land u \notin t(ptr\Delta) \land a \in F \land \#_a u < \#_b t : ub \in t(ptr\Delta)).
\]

end of definition

The definition reflects that a component accepts every communication signal it might receive.

4.1 Delay-insensitive enclosure

The *delay-insensitive enclosure* of a component is the directed trace structure that is associated with the maximal delay-insensitive communication in a channel, such that no computation hazard occurs.
definition
For component $\Gamma$ and i/o-direction $F$, such that $F$ is the i/o-direction of $\Gamma$, the delay-insensitive enclosure of $\Gamma$, denoted by $\text{dec}\Gamma$, is defined recursively by

(i) \[ a(\text{dec}\Gamma) \stackrel{\text{def}}{=} a(\text{ptr}\Gamma), \]

(ii) \[ d(\text{dec}\Gamma) \stackrel{\text{def}}{=} d(\text{ptr}\Gamma), \]

(iii) \[ t(\text{dec}\Gamma) \stackrel{\text{def}}{=} \{ x, a : x \in t(\text{dec}\Gamma) \] 
\[ \quad \wedge (a \in o\Gamma \Rightarrow (E w : w \in t(\text{ptr}\Gamma) \wedge wcFx : \#_aw > \#_ax)) \]
\[ \quad \wedge (a \in i\Gamma \Rightarrow (A y, b : y \in t(\text{ptr}\Gamma) \wedge b \in i\Gamma \wedge ycFxa \wedge \#_by < \#_bxa \] 
\[ \quad \quad : yb \in t(\text{ptr}\Gamma) \] 
\[ \quad \}) \cup \{ e \} , \]

(iv) completeness axiom: $t(\text{dec}\Gamma)$ contains no traces that are not required by (iii).

end of definition

Due to the definition of delay-insensitive enclosure and the nonemptyness and prefix-closedness of $\text{ptr}$ we get:

property
For component $\Gamma$,
\[ \text{dec}\Gamma \] is nonempty and prefix-closed.

end of property

Absence of computation hazard is reflected by the property:

property
Given component $\Gamma$. Let $\Delta$ such that $i\Delta = o\Gamma$, $o\Delta = i\Gamma$, and $\text{ptr}\Delta = \text{dec}\Gamma$.
\[ \Gamma \text{ and } \Delta \text{ have no computation hazard.} \]

end of property

The delay-insensitivity of the communication in the channel, to which the delay-insensitive enclosure of a component is associated, is reflected by:

property
For component $\Gamma$,
\[ \text{dec}\Gamma \in D_4 \]

end of property
The maximality of the delay-insensitive enclosure follows from: suppose that we add a trace to \( t(\text{dec}\Gamma) \) for some component \( \Gamma \). Due to the prefix-closedness requirement we add a trace \( sa \), such that \( s \in t(\text{dec}\Gamma) \), \( a \in a(\text{dec}\Gamma) \), and \( sa \in t(\text{dec}\Gamma) \). If \( a \) is an input of \( \Gamma \), then the addition of \( sa \) to \( t(\text{dec}\Gamma) \) destroys the absence of computation hazard. If \( a \) is an output of \( \Gamma \), then there exists no trace in \( t(ptr\Gamma) \), that is composable under the i/o-direction of \( \Gamma \) with \( sa \); hence, trace \( sa \) is not associated with a sequence of instances of events, that can occur in the channel when \( \Gamma \) communicates delay-insensitively: that is why \( sa \notin t(\text{dec}\Gamma) \).

We present some more properties:

property

For component \( \Gamma \) that communicates delay-insensitively and i/o-direction \( F \), such that \( F \) is the i/o-direction of \( \Gamma \),

\[
(A u : u \in t(\text{dec}\Gamma) : (E t : t \in t(ptr\Gamma) : t \in u)).
\]

end of property

property

For component \( \Gamma \),

\[ ptr\Gamma \in D_s \Rightarrow (\text{dec}\Gamma = ptr\Gamma). \]

end of property

In general, \( \text{dec} \) is not monotonic, see the following example.

example

We consider components \( \Gamma \) and \( \Delta \). Their communication behavior is given by:

\[
\begin{align*}
\text{i} \Gamma & \overset{\text{def}}{=} \{ a \}, & \text{o} \Gamma & \overset{\text{def}}{=} \{ b \}, \\
\text{i} \Delta & \overset{\text{def}}{=} \{ a \}, & \text{o} \Delta & \overset{\text{def}}{=} \{ b \}, \\
\text{ptr} \Gamma & \overset{\text{def}}{=} \langle \{ a, b \}, \{ a, ab \}, \{ a \} \oplus \{ b \} \rangle, & \text{ptr} \Delta & \overset{\text{def}}{=} \langle \{ a, b \}, \{ a, ab \}, \{ a \} \oplus \{ b \} \rangle.
\end{align*}
\]

For the delay-insensitive enclosures we find:

\[
\text{dec}\Gamma = \text{ptr}\Gamma, \text{ and }
\text{dec}\Delta = \langle \{ a, b \}, \{ a \oplus \{ b \} \rangle.
\]

We see that \( \text{ptr}\Gamma \subseteq \text{ptr}\Delta \), but \( \neg (\text{dec}\Gamma \subseteq \text{dec}\Delta) \).

end of example

However, the following property shows some relation between the delay-insensitive enclosures of components.
4.2 The d-i communication behavior of a component

In this subsection we are interested in the impact of delay-insensitive communication on the communication behavior of a component. We define the d-i communication behavior of a component that communicates delay-insensitively, i.e. the maximal communication behavior of the component at the ports of the component when the communication in the channel, to which the component is connected, is delay-insensitive.

definition
For component $\Gamma$ that communicates delay-insensitively and i/o-direction $F$, such that $F$ is the i/o-direction of $\Gamma$, the directed trace structure that is associated with the d-i communication behavior of $\Gamma$, which is denoted by $\text{ebdi}\Gamma$, is defined by

$$
\text{ebdi}\Gamma \overset{\text{def}}{=} \langle a(\text{ptr}\Gamma), \{t, u : t \in t(\text{ptr}\Gamma) \land tC u \land u \in t(\text{dec}\Gamma) : t\}, d(\text{ptr}\Gamma) \rangle.
$$
end of definition

The definition above reflects that only the traces in $t(\text{ptr}\Gamma)$, that are composable with some trace in $t(\text{dec}\Gamma)$, are associated with the d-i communication behavior of component $\Gamma$.

property
For component $\Gamma$ and i/o-direction $F$, such that $F$ is the i/o-direction of $\Gamma$,

$$( A t : t \in t(\text{ptr}\Gamma) \land (E u : u \in t(\text{dec}\Gamma) : tC u) : t \in t(\text{dec}\Gamma)) .$$
end of property

From the property above we derive for the d-i communication behavior of a component that communicates delay-insensitively:

property
For a component $\Gamma$ that communicates delay-insensitively,

$$\text{ebdi}\Gamma = \text{ptr}\Gamma \cap \text{dec}\Gamma .$$
end of property

Due to the nonemptyness and prefix-closedness of $\text{ptr}$ and $\text{dec}$ we get:
Notes on Delay-Insensitive Communication

property
For component r,
\( \text{cbdir} \) is nonempty and prefix-closed.
end of property

We rewrite a property of subsection 4.1 into:

property
For components r and A, such that \( \text{cbdir} \subseteq \text{ptr}A \) and \( \text{ptr}A \subseteq \text{ptr}r \),
\[ \text{dec}r = \text{dec}A. \]
end of property

5 Implementation of specifications

Throughout this section we shall consider two components. One is called the environment, denoted by \( \Phi \), the other is called the module, denoted by \( \Xi \), see figure 7.

![Environment and module](image)

Environment \( \Phi \) and module \( \Xi \).

Given is the communication behavior of \( \Phi \), which we consider as a specification that should be implemented by module \( \Xi \). We assume that the communication between \( \Xi \) and \( \Phi \) is delay-insensitive. One requirement to call module \( \Xi \) an implementation that fulfills environment \( \Phi \) is that \( \Xi \) and \( \Phi \) have no computation hazard. Furthermore, the communication in the channel between \( \Xi \) and \( \Phi \) should have no transmission hazard. These are, however, not sufficient: we do not want the communication between \( \Xi \) and \( \Phi \) to end, unless \( \Phi \) is not 'interested' in using \( \Xi \) any more. This requirement we call
"module Ξ and environment Φ have no unspecified termination hazard".

5.0 Ways of interpreting directed trace structures

Until now, we have only considered lazy interpretations of directed trace structures associated with a component, i.e. a component does not engage in a sequence of instances of events if the corresponding trace of symbols is not in the trace set; a component has an obligation to accept inputs, but it has no obligation to send outputs, see subsection 2.2.1.

We introduce a greedy interpretation of directed trace structures associated with a component, in particular a module: whenever a component is, according to its directed trace structure, allowed to send an instance of an output event, it will send an instance of some output event unless an instance of some input event occurs first.

When we refer to lazy component or greedy component, we mean that we interpret the directed trace structure of the component in a lazy way or a greedy way, respectively.

example

Let

\[
\begin{align*}
\Xi & \overset{\text{def}}{=} \emptyset, & \Omega & \overset{\text{def}}{=} \{a\}, \\
0_\Phi & \overset{\text{def}}{=} \emptyset, & 1_\Phi & \overset{\text{def}}{=} \{a\}, \\
ptr_\Xi & \overset{\text{def}}{=} \langle \{a\}, \{e\}, \{a\} \odot \emptyset \rangle, & ptr_\Phi & \overset{\text{def}}{=} \langle \{a\}, \{e\}, \{a\} \odot \emptyset \rangle.
\end{align*}
\]

When Ξ is greedy, then the environment Φ can wait for an instance of the communication signal: it will be sent by Ξ; hence, it will arrive at the port of Φ.

end of example

In section 3 we have argued that distinguishing between "communication in a channel" and "communication behavior of a component" enables us to discuss unspecified termination hazard. Unspecified termination hazard is related to "livelock" and "deadlock", cf. [Kimura, Kaldewaij].

If a component, say r is greedy, i.e. we interpret ptrr in a greedy way, then we are allowed to interpret ebdr in a greedy way, since ebdr(;;ptrr; however, the following example illustrates that we are not allowed to interpret decr in a greedy way.

example

For component Γ, such that

\[
\begin{align*}
i_\Gamma & \overset{\text{def}}{=} \{a, b\}, & o_\Gamma & \overset{\text{def}}{=} \{c\}, \\
t(ptr_\Gamma) & \overset{\text{def}}{=} \text{pref} \{abc, ba\},
\end{align*}
\]

we notice

\[
t(\text{decr}) = \text{pref} \{abc, bac\}.
\]
Let $\alpha$, $\beta$, and $\gamma$ be the instances of the events (at the ports of the environment of $\Gamma$) to which, respectively, $a$, $b$, and $c$ are associated. In despite of $\Gamma$ being greedy, the environment of $\Gamma$ should not wait for $\gamma$ after it has sent $\alpha$ and $\beta$: it might not occur.

end of example

In order to discuss a liveness property, we need that at least one of the components is greedy. It seems reasonable to require a module to behave in such a way that its communication behavior is modeled by the greedy interpretation of its directed trace structure.

In subsection 5.1 we consider a lazy environment and a greedy module. In subsection 5.2 we consider an environment and a module that are both greedy.

By using these lazy and greedy interpretations we are not able to express all liveness properties, e.g. we cannot express fairness properties. It is possible to consider more refined interpretations of directed trace structures, e.g. by introducing refusal sets, cf. [Hoare, VerhoeffPC], or introducing infinite traces, cf. [Van Horn, Black]. By introducing infinite traces one is able to deal with fairness properties.

5.1 Unspecified termination for lazy specification

In this subsection we deal with unspecified termination hazard between a greedy module and a lazy environment.

definition

For module $\Xi$ and environment $\Phi$ and i/o-direction $F$, such that $i\Xi = o\Phi$, $o\Xi = i\Phi$, and $F$ is the i/o-direction of $\Xi$, we say that module $\Xi$ and lazy environment $\Phi$ have no unspecified termination hazard, if and only if

$$\forall a, t, u : t \in t(\text{cbdi}\Xi) \land iF u \land u \in t(\text{cbdi}\Phi) \land a \in i\Phi \land u \in t(\text{cbdi}\Phi) \land (\text{bag} t = \text{bag} u) : (\exists b : b \in o\Xi : t b \in t(\text{cbdi}\Xi)),$$

end of definition

We explain this definition: the passages between hyphens refer to this definition. We consider the communication behaviors of $\Xi$ and $\Phi$ at some moment - cf. $t \in t(\text{cbdi}\Xi) \land iF u \land u \in t(\text{cbdi}\Phi)$ . $\Phi$ is able to accept (under the assumption of delay-insensitive communication) an input - cf. $a \in i\Phi \land u \in t(\text{cbdi}\Phi)$ and no communication signal is traveling along a path between $\Xi$ and $\Phi$ - cf. $\text{bag} t = \text{bag} u$ . Now, $\Xi$ should produce an output - cf. $b \in o\Xi$, and $t b \in t(\text{cbdi}\Xi)$ . Notice that, if $\Xi$ and $\Phi$ have no computation hazard, this output will be accepted by $\Phi$.

Due to the prefix-closedness of $\text{cbdi}\Phi$ we find:
property

For module \( \Xi \) and environment \( \Phi \) and i/o-direction \( F \), such that \( \Xi = \alpha F \), \( \alpha F \) = \( \iota \Phi \), and \( F \) is the i/o-direction of \( \Xi \), module \( \Xi \) and lazy environment \( \Phi \) have no unspecified termination hazard, if and only if

\[
(A \ a \ , \ t \ , \ u : t \in (\text{cbdi} \Xi) \land \tau F u \land a \in \iota \Phi \land u a \in t(\text{cbdi} \Phi) \land (\text{bag} t = \text{bag} u) \\
\quad : (E b : b \in \alpha \Xi : t b \in (\text{cbdi} \Xi))
\]

end of property

5.2 Unspecified termination for greedy specification

In this subsection we deal with unspecified termination hazard between a greedy module and a greedy environment. The environment being greedy leads to a weaker definition of absence of unspecified termination hazard than in the case of a lazy environment.

definition

For module \( \Xi \) and environment \( \Phi \) and i/o-direction \( F \), such that \( \Xi = \alpha F \), \( \alpha F \) = \( \iota \Phi \), and \( F \) is the i/o-direction of \( \Xi \), we say that module \( \Xi \) and greedy environment \( \Phi \) have no unspecified termination hazard, if and only if

\[
(A \ a \ , \ t \ , \ u : t \in (\text{cbdi} \Xi) \land \tau F u \land a \in \iota \Phi \land u a \in t(\text{cbdi} \Phi) \land (\text{bag} t = \text{bag} u) \\
\quad \land (A c : uc \in t(\text{cbdi} \Phi) : c \in \iota F ) \\
\quad : (E b : b \in \alpha \Xi : t b \in (\text{cbdi} \Xi))
\]

end of definition

We explain this definition: the passages between hyphens refer to this definition. Notice that, since \( \Phi \) is greedy, the communication between \( \Xi \) and \( \Phi \) does not terminate when \( \Phi \) is able to send an output. We consider the communication behaviors of \( \Xi \) and \( \Phi \) at some moment - cf. \( t \in (\text{cbdi} \Xi) \land \tau F u \land u \in t(\text{cbdi} \Phi) \) -. \( \Phi \) is able to accept (under the assumption of delay-insensitive communication) an input - cf. \( a \in \iota \Phi \land u a \in t(\text{cbdi} \Phi) \) -. \( \Phi \) is not able to send any output - cf. \( (A c : uc \in t(\text{cbdi} \Phi) : c \in \iota F ) \) -, and no communication signal is traveling along a path between \( \Xi \) and \( \Phi \) - cf. \( \text{bag} t = \text{bag} u \) -. Now, \( \Xi \) should produce an output - cf. \( b \in \alpha \Xi \), and \( t b \in (\text{cbdi} \Xi) \) -. Notice that, if \( \Xi \) and \( \Phi \) have no computation hazard, this output will be accepted by \( \Phi \).

Again we find due to the prefix-closedness of \( \text{cbdi} \Phi \):

property

For module \( \Xi \) and environment \( \Phi \) and i/o-direction \( F \), such that \( \Xi = \alpha F \), \( \alpha F \) = \( \iota \Phi \), and \( F \) is the i/o-direction of \( \Xi \), module \( \Xi \) and greedy environment \( \Phi \) have no unspecified termination hazard, if and only if
6 A consideration

The usage of the direction of directed trace structures is awkward. In this monograph it has only been used in the definitions of \( D_4 \) and \( C_4 \). In section 2 we explained that we use directed trace structures to model either the communication in a channel at some points or the communication behavior of a component at its ports, see figure 8.

In order not to distract the attention from these points or ports, in figure 8 we do not show the rest of the channel or the component. The instances of events that occur at these points or ports are occurrences of events at these points or ports. When we are interested in the occurrences of events at some points or ports, we are not able to distinguish any directions: directions are issues that are related to the usage of the paths in a channel or the usage of the ports by a component. Hence, directions are interpretational issues! On this consideration we base our intention to use in future (undirected) trace structures to model (delay-insensitive) communication.
Acknowledgements

The author thanks the members of the Institute for Biomedical Computing of the Washington University at St. Louis for their contributions to his understanding of the presented material; in particular the intense and enthusiastic collaboration from Charles Molnar has sharpened many ideas and has been very stimulating during the development of the presented theory. Furthermore, the author gratefully acknowledges the discussions with and suggestions from Martin Rem, Ting-Pien Fang, Jan Tijmen Udding, Alain Martin, Jan van de Snepscheut, Jo Ebergen, and Tom Verhoeff. Thanks also go to the members of the Department of Computer Science of the Washington University, particularly Jerome R. Cox jr. and Takayuki (Dan) Kimura, and to the members of the Eindhoven VLSI club for their criticisms on this material.

References

[Barney]

[Black]

[Burstyn]
David Burstyn, *State-Graph Based Verification Techniques for Delay-Insensitive Modules*, Technical Memorandum No. 318, Computer Systems Laboratory, Institute for Biomedical Computing, Washington University, St. Louis, Missouri.

[Chaney-Molnar]

[Dill-Clarke]

[Ebergen]
Notes on Delay-Insensitive Communication


Notes on Delay-Insensitive Communication


[Martin86]

[Mazurkiewicz]

[Mead-Conway]

[Miller]

[Molnar]

[Molnar-Fang]

[Molnar-Fang-Rosenberger]

[Muller-Bartky]

[Rem]
[Rem-van de Snepscheut-Udding]

[Rosenberger]

[Schols]

[ScholsIPa]

[ScholsIPb]

[Seitz79]

[Seitz80]

[Siccama]

[Udding]

[Unger]
[van de Snepscheut]

[Van Horn]

[Verhoeff]

[VerhoeffPC]
Tom Verhoeff, private communication.

[Yakovlev]
In this series appeared:

<table>
<thead>
<tr>
<th>No.</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>85/01</td>
<td>R.H. Mak</td>
<td>The formal specification and derivation of CMOS-circuits</td>
</tr>
<tr>
<td>85/02</td>
<td>W.M.C.J. van Overveld</td>
<td>On arithmetic operations with M-out-of-N-codes</td>
</tr>
<tr>
<td>85/03</td>
<td>W.J.M. Lemmens</td>
<td>Use of a computer for evaluation of flow films</td>
</tr>
<tr>
<td>85/04</td>
<td>T. Verhoeff, H.M.J.L. Schols</td>
<td>Delay insensitive directed trace structures satisfy the foam rubber wrapper postulate</td>
</tr>
<tr>
<td>86/01</td>
<td>R. Koymans</td>
<td>Specifying message passing and real-time systems</td>
</tr>
<tr>
<td>86/02</td>
<td>G.A. Bussing, K.M. van Hee, M. Voorhoeve</td>
<td>ELISA, A language for formal specifications of information systems</td>
</tr>
<tr>
<td>86/03</td>
<td>Rob Hoogerwoord</td>
<td>Some reflections on the implementation of trace structures</td>
</tr>
<tr>
<td>86/04</td>
<td>G.J. Houben, J. Paredaens, K.M. van Hee</td>
<td>The partition of an information system in several parallel systems</td>
</tr>
<tr>
<td>86/05</td>
<td>Jan L.G. Dietz, Kees M. van Hee</td>
<td>A framework for the conceptual modeling of discrete dynamic systems</td>
</tr>
<tr>
<td>86/06</td>
<td>Tom Verhoeff</td>
<td>Nondeterminism and divergence created by concealment in CSP</td>
</tr>
<tr>
<td>86/07</td>
<td>R. Gerth, L. Shira</td>
<td>On proving communication closedness of distributed layers</td>
</tr>
<tr>
<td>86/09</td>
<td>C. Huizing, R. Gerth, W.P. de Roever</td>
<td>Full abstraction of a real-time denotational semantics for an OCCAM-like language</td>
</tr>
<tr>
<td>86/10</td>
<td>J. Hooman</td>
<td>A compositional proof theory for real-time distributed message passing</td>
</tr>
<tr>
<td>86/11</td>
<td>W.P. de Roever</td>
<td>Questions to Robin Milner - A responder’s commentary (IFIP86)</td>
</tr>
<tr>
<td>86/12</td>
<td>A. Boucher, R. Gerth</td>
<td>A timed failures model for extended communicating processes</td>
</tr>
</tbody>
</table>
86/13 R. Gerth
W.P. de Roever

Proving monitors revisited: a first step towards verifying object oriented systems (Fund. Informatica IX-4)

86/14 R. Koymans

Specifying passing systems requires extending temporal logic

87/01 R. Gerth

On the existence of sound and complete axiomatizations of the monitor concept

87/02 Simon J. Klaver
Chris F.M. Verberne

Federatieve Databases

87/03 G.J. Houben
J.Paredaens

A formal approach to distributed information systems

87/04 T.Verhoeff

Delay-insensitive codes - An overview

87/05 R.Kuiper

Enforcing non-determinism via linear time temporal logic specification.

87/06 R.Koymans

Temporele logica specificatie van message passing en real-time systemen (in Dutch).

87/07 R.Koymans

Specifying message passing and real-time systems with real-time temporal logic.

87/08 H.M.J.L. Schols

The maximum number of states after projection.

87/09 J. Kalisvaart
L.R.A. Kessener
W.J.M. Lemmens
M.L.P. van Lierop
F.J. Peters
H.M.M. van de Wetering

Language extensions to study structures for raster graphics.

87/10 T.Verhoeff

Three families of maximally nondeterministic automata.

87/11 P.Lemmens

Eldorado ins and outs. Specifications of a data base management toolkit according to the functional model.

87/12 K.M. van Hee and A.Lapinski

OR and AI approaches to decision support systems.

87/13 J.C.S.P. van der Woude

Playing with patterns, searching for strings.

87/14 J. Hooman

A compositional proof system for an occam-like real-time language.
<table>
<thead>
<tr>
<th>Volume</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>87/15</td>
<td>C. Huizing, R. Gerth, W.P. de Roever</td>
<td>A compositional semantics for statecharts</td>
</tr>
<tr>
<td>87/16</td>
<td>H.M.M. ten Eikelder, J.C.F. Wilmont</td>
<td>Normal forms for a class of formulas</td>
</tr>
<tr>
<td>87/17</td>
<td>K.M. van Hee, G.-J. Houben, J.L.G. Dietz</td>
<td>Modelling of discrete dynamic systems framework and examples</td>
</tr>
<tr>
<td>87/18</td>
<td>C.W.A.M. van Overveld</td>
<td>An integer algorithm for rendering curved surfaces</td>
</tr>
<tr>
<td>87/19</td>
<td>A.J. Seebregts</td>
<td>Optimalisering van file allocatie in gedistribueerde database systemen</td>
</tr>
<tr>
<td>87/20</td>
<td>G.J. Houben, J. Paredaens</td>
<td>The $R^2$-Algebra: An extension of an algebra for nested relations</td>
</tr>
<tr>
<td>87/21</td>
<td>R. Gerth, M. Codish, Y. Lichtenstein, E. Shapiro</td>
<td>Fully abstract denotational semantics for concurrent PROLOG</td>
</tr>
<tr>
<td>88/01</td>
<td>T. Verhoeff</td>
<td>A Parallel Program That Generates the Möbius Sequence</td>
</tr>
<tr>
<td>88/02</td>
<td>K.M. van Hee, G.J. Houben, L.J. Somers, M. Voorhoeve</td>
<td>Executable Specification for Information Systems</td>
</tr>
<tr>
<td>88/03</td>
<td>T. Verhoeff</td>
<td>Settling a Question about Pythagorean Triples</td>
</tr>
<tr>
<td>88/04</td>
<td>G.J. Houben, J. Paredaens, D. Tahon</td>
<td>The Nested Relational Algebra: A Tool to handle Structured Information</td>
</tr>
<tr>
<td>88/05</td>
<td>K.M. van Hee, G.J. Houben, L.J. Somers, M. Voorhoeve</td>
<td>Executable Specifications for Information Systems</td>
</tr>
<tr>
<td>88/06</td>
<td>H.M.J.L. Schols</td>
<td>Notes on Delay-Insensitive Communication</td>
</tr>
<tr>
<td>88/07</td>
<td>C. Huizing, R. Gerth, W.P. de Roever</td>
<td>Modelling Statecharts behaviour in a fully abstract way</td>
</tr>
<tr>
<td>88/08</td>
<td>K.M. van Hee, G.J. Houben, L.J. Somers, M. Voorhoeve</td>
<td>A Formal model for System Specification</td>
</tr>
<tr>
<td>88/09</td>
<td>A.T.M. Aerts, K.M. van Hee</td>
<td>A Tutorial for Data Modelling</td>
</tr>
</tbody>
</table>
A Formal Approach to Designing Delay Insensitive Circuits.