Characterizations of delay-insensitive communication protocols

Citation for published version (APA):

Document status and date:
Published: 01/01/1989

Publisher 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.
Characterizations of Delay-Insensitive Communication Protocols

by

Tom Verhoeff

May, 1989
This is a series of notes of the Computing
Science Section of the Department of
Mathematics and Computing Science
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.
Characterizations of Delay-Insensitive Communication Protocols

Tom Verhoeff
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
NL-5600 MB EINDHOVEN
The Netherlands
wstomv@eutws1.uucp
wsintom@heitue5.bitnet

May 1989

Abstract
This paper studies protocols for asynchronous communication over an interface consisting of unidirectional channels. In asynchronous communication, the sender can initiate a transmission without cooperation of the receiver. Contrasting with traditional data-flow networks, the channels that we consider do not synchronize at the receiving end: messages, once on their way, are delivered regardless of the readiness of the receiver to accept them. The situation where a message is delivered to an unready receiver is called computation interference. A protocol is said to be delay-insensitive when it can be guaranteed—without making assumptions about propagation delays—that computation interference cannot occur. We give several characterizations of delay-insensitive protocols and a new proof for the Fundamental Characterization Theorem. The emphasis is on the mathematical treatment of the concepts involved.

0 Introduction and Overview
We begin by giving a physical motivation for our investigation. Consider a digital circuit connected to its environment by an interface consisting of conducting wires. In the digital mode of operation, circuit and environment communicate by exchanging discrete voltage transitions. A voltage transition, once initiated, propagates along a wire to the receiver. The receiving end, however, need not always be ready to process an incoming transition.
The reason for this is that the incoming transition may violate the assumptions underlying the digital mode of operation.

For example, an OR-gate in the stable state with one input low, the other input high, and—consequently—the output also high, cannot properly process transitions on both inputs "simultaneously". The best that can happen when both inputs change is that the output remains high or produces a pair of proper transitions (high to low and back to high). But it is also possible that a non-digital pulse ensues\(^9\). What actually happens when both inputs change depends intricately on the (relative) timing of the transitions and the physical structure of the circuitry.

We are interested in communication protocols that guarantee adherence to the digital mode assumptions even when no assumptions are made about the propagation delays incurred in the connecting wires. Such protocols are called delay-insensitive. In order to define and investigate such protocols we introduce a formal model for asynchronous two-party communication. The operational semantics is given in terms of a transition system. The model includes the possibility to specify under what circumstances a party is ready to accept which messages. Our model formalizes the Foam Rubber Wrapper Postulate put forward in [3] as an approach to define delay-insensitivity. We are aware of the gap between circuit physics on the one hand and transition-system semantics on the other. It is not the validity of this type of semantics for circuits that we wish to question here.

In Section 1 we present the formal communication model, define our notion of delay-insensitivity, and state the Fundamental Characterization Theorem. Sections 2, 3, and 4 introduce auxiliary concepts and prove characterizations based on these concepts. Together they constitute a new proof for the Fundamental Characterization Theorem. Finally, Section 5 summarizes the results and mentions some relationships with other work.

1 Delay-Insensitivity and the JTU-Rules

We start with the introduction of some terminology and notations. The two communicating parties are referred to as Module and Environment. Let \( I \) and \( O \) be disjoint sets of symbols, identifying the channels in the interface. The direction of symbols in \( I \) and \( O \) is said to be input and output respectively. The directions are to be interpreted with respect to Module. The sets \( I \) and \( O \) are fixed for the remainder of the paper. We denote their union by \( A \). Variables \( a \) through \( d \) range over \( A \). A trace is a member of \( A^* \), i.e. a finite-length sequence of symbols from \( A \). It records a communication history at one side of the interface. The contents of the messages communicated is irrelevant for our problem. Hence, there is only the need to record the occurrence of a communication action, for which we employ the symbol identifying the channel involved in that communication action. Variables \( s \) through \( z \) range over \( A^* \). The empty trace is denoted by \( e \) and concatenation of traces is denoted by juxtaposition. Trace \( t \) is a prefix of trace \( s \) when \((\exists u :: tu = s)\). Subset \( T \) of \( A^* \) is called prefix-closed when

\(^9\)In fact, the pair of proper transitions may also degrade into a non-digital pulse when propagated along the output wire, if no special precautions are taken.
(∀ s, t : s ∈ T ∧ t prefix of s : t ∈ T).

The length of trace t is denoted by ℓ(t). The symbol bag of trace t is denoted by t#, that is, t# is a mapping from A into the natural numbers such that t#a (the result of applying t# to a) is the number of occurrences of symbol a in trace t.

A protocol specification, or specification for short, is a non-empty prefix-closed subset of A*. It gives the set of communication histories allowed at both ends of the interface. We define its operational semantics as a transition system. A transition system is a triple (Q, q, →), where Q is some set of states, q ∈ Q is the initial state, and → ⊆ Q × Q is the transition relation. The transition system associated with specification T is

\[ (A^* × A^*, (ε, ε), →) \]

where → is the smallest relation such that

\[
\begin{align*}
(t, u) &\rightarrow (ta, u) \quad \text{if} \quad a ∈ O \land ta ∈ T \\
(t, u) &\rightarrow (t, ua) \quad \text{if} \quad a ∈ I \land ua ∈ T \\
(t, u) &\rightarrow (ta, u) \quad \text{if} \quad a ∈ I \land t#a < u#a \\
(t, u) &\rightarrow (t, ua) \quad \text{if} \quad a ∈ O \land u#a < t#a
\end{align*}
\]

(transmissions)

(receptions)

Thus, a state is a pair of traces over A, and in the initial state both traces of the pair are empty. The left component of the pair can be thought of as the local state of Module while the right component is associated with Environment. Outputs travel from Module to Environment, inputs from Environment to Module.

The first transition rule expresses a state change where Module extends its local state with an output symbol if the resulting local state belongs to the specification. Similarly, the second transition rule expresses a state change where Environment’s local state is extended with an input symbol (which acts as a transmission initiated by Environment). Hence, transmissions will only be initiated if they are in agreement with the specification. The third transition rule expresses a state change where Module’s local state is extended with an input symbol if that symbol was sent more often (by Environment) than received so far (by Module), i.e. if a message was on its way over the channel identified by that symbol. Similarly, the fourth transition rule expresses a state change where Environment receives an output (sent earlier by Module). Hence, a reception takes place only if the corresponding transmission precedes it. Note, however, that at this stage receptions are not required to obey the specification.

We call state (t, u) reachable under specification T when it can be reached from the initial state via zero or more T-transitions, that is, when

\[ (ε, ε) \rightarrow^* (t, u) \]

where →* denotes the transitive and reflexive closure of →. The set of states reachable under T is denoted by rT. State (t, u) is called safe under T when (t, u) ∈ T × T, that is, when both local states belong to the specification. Specification T is delay-insensitive, or DI for short, when
that is, when all reachable states are safe. The situation where a reachable state is not in agreement with the specification is called \textit{computation interference}. It corresponds to a possible violation of the digital mode assumptions. The central problem of this paper is the characterization of delay-insensitivity.

We give three examples to illustrate these definitions.

\textbf{Example 0} Consider the specification $T = \{\varepsilon\}$ for arbitrary $I$ and $O$. Its transition relation is empty. The initial state is the only reachable state and, hence, $rT = T \times T$. Consequently, specification $T$ is delay-insensitive.

\textbf{Example 1} Assume $I = \{a,b\}$ with $a \neq b$ and $O = \emptyset$. Now consider specification $T = \{\varepsilon,a,ab\}$. For $T$'s transition system we give two (of the three) maximal transition sequences starting in the initial state. The first sequence consists of two transmissions interleaved with the corresponding receptions:

$$(\varepsilon, \varepsilon) \xrightarrow{T} (\varepsilon, a) \xrightarrow{T} (a, a) \xrightarrow{T} (a, ab) \xrightarrow{T} (ab, ab).$$

Thus, $(ab, ab) \in rT$. Notice that the final state $(ab, ab)$ is safe. The second sequence consists of two transmissions followed by two receptions, where the order of the receptions differs from the order of the corresponding transmissions:

$$(\varepsilon, \varepsilon) \xrightarrow{T} (\varepsilon, a) \xrightarrow{T} (\varepsilon, ab) \xrightarrow{T} (b, ab) \xrightarrow{T} (ba, ab).$$

Thus, $(ba, ab) \in rT$, but now the final state $(ba, ab)$ is not safe, i.e. there is computation interference. In fact, the intermediate state $(b, ab)$ was already not safe. Therefore, the specification $T$ is not delay-insensitive.

This example also exhibits one of the complications inherent to asynchronous communication: Even if there are no messages on their way, then the local states of the communicating parties may differ.

\textbf{Example 2} Assuming $I = \{a\}$ and $O = \{b\}$ define specification $T$ by

$$T = \{t \mid (\forall s : s \text{ prefix of } t : 0 \leq t\#a - t\#b \leq 1)\}.$$ 

Thus, $T$ consists of all traces in which symbols $a$ and $b$ alternate and which do not start with $b$:

$$(\varepsilon, a, ab, aba, abab, ababa, \ldots).$$

Specification $T$ describes a two-phase protocol. In each state, exactly one transition is possible. Hence, there exists only one maximal transition sequence, which is infinite and starts out

$$(\varepsilon, \varepsilon) \xrightarrow{T} (\varepsilon, a) \xrightarrow{T} (a, a) \xrightarrow{T} (ab, a) \xrightarrow{T} (ab, ab) \xrightarrow{T} \ldots.$$
The two-phase protocol $T$ is delay-insensitive, because all reachable states are safe. 

In [4, 5] Udding gives a characterization of delay-insensitive specifications. He also deals with transmission interference—which can occur when a channel carries more than one message—but we will ignore that here: our channels have unbounded buffering capacity. Udding defines the following predicates on specifications. We adhere to the names given in [4].

Specification $T$ satisfies Rule $R_3$ (called $R_1$ in [5]) when for all traces $s$ and $t$, and symbols $a$ and $b$ of the same direction we have

$$sabt \in T \equiv sbat \in T.$$  

(4)

Specification $T$ satisfies Rule $R''_3$ (called $R'_2$ in [5]) when for all traces $s$ and $t$, and symbols $a$, $b$, and $c$ such that the direction of $a$ and $c$ differs from the direction of $b$, we have

$$sabc \in T \land sbat \in T \Rightarrow sbatc \in T.$$  

(5)

Specification $T$ satisfies Rule $R''_4$ (called $R''_2$ in [5]) when for all traces $s$ and symbols $a$ and $b$ of different direction we have

$$sa \in T \land sb \in T \Rightarrow sab \in T.$$  

(6)

Remark for the curious: Rules $R_0$ and $R_1$ of [4] were already incorporated in our notion of a specification. Rule $R_2$ deals with transmission interference, which we decided to ignore here.

We say that a specification satisfies the JTU-Rules when it satisfies Rules $R_3$, $R''_3$, and $R''_4$. Notice that the specifications of Examples 0 and 2 trivially satisfy the JTU-Rules. The specification of Example 1, however, satisfies Rules $R''_3$ and $R''_4$ but not Rule $R_3$.

This paper is centered around the following theorem.

**Theorem 0** (Fundamental Characterization Theorem of Delay-Insensitivity) Specification $T$ is DI if and only if it satisfies the JTU-Rules. 

The implication from right to left is "hard" and was—in a slightly more general form—first stated and proved in [4, Thm. 4.1]. Several attempts at simplifying the proof have failed. In this paper we present, what we believe to be, a simple proof. In a sense, the justification of the JTU-Rules as given in [4] constitutes an informal proof of the implication from left to right. This is the "easy" part. A formal proof of this part can also be found in [7].

---

1Note on terminology: In [4] Udding defines delay-insensitivity directly in terms of the JTU-Rules and he shows that it implies absence of computation interference, which he defines as (8) below.
2 Composability and Convexity

In this section we present three—fairly straightforward—characterizations of delay-insensitivity (Theorems 1, 2, and 3). They do not get very far in bridging the gap between delay-insensitivity and the JTU-Rules, but they are useful nonetheless, since they take us away from the definition of delay-insensitivity in terms of the operational transition system.

We start by noting the following symmetry in the transition system $S$ associated with specification $T$. The transition system obtained from $S$ by exchanging left and right components of states equals the transition system associated with $T$ when the roles of $I$ and $O$ are exchanged. This symmetry will be referred to as $I/O$-symmetry. Also notice that the JTU-Rules are $I/O$-symmetric, since they involve (in)equality of direction only.

The first characterization (Theorem 1 below) is based on the observation that the initial state is safe and that transmission transitions do not disturb safety by definition. Therefore, all reachable states are safe and only if each reception transition from a safe reachable state leads to a safe state. We have also incorporated some knowledge about reachable states viz. that the number of receptions of a symbol cannot exceed the number of its transmissions. This is formally expressed in

Property 0 For specification $T$ and state $(t, u) \in rT$ we have

$$(\forall a : a \in I : t\#a \leq u\#a) \land (\forall a : a \in O : t\#a \geq u\#a). \quad (7)$$

Therefore, if $(t, u) \in rT$ and $t\#a < u\#a$, then $a \notin O$ and, hence, $a \in I$.

Theorem 1 Specification $T$ is DI if and only if

$$(\forall t, u, a : (t, u) \in rT \cap (T \times T) : (t\#a < u\#a \Rightarrow ta \in T) \land (u\#a < t\#a \Rightarrow ua \in T)). \quad (8)$$

Proof

Only if: Assuming $T$ is DI we show (8). Let $(t, u) \in rT$ be such that $t \in T$ and $u \in T$. We derive

$t\#a < u\#a \\
\Rightarrow \{ \text{Property 0} \} \\
a \in I \land t\#a < u\#a \\
\Rightarrow \{ \text{definition of } \rightarrow \} \\
(t, u) \rightarrow^T (ta, u) \\
\Rightarrow \{ \text{definition of } rT, \text{ using } (t, u) \in rT \} \\
(ta, u) \in rT$
\[ (ta, u) \in (T \times T) \]
\[ \Rightarrow \{ \text{set calculus} \} \]
\[ ta \in T \]

The conjunct \( u\#a < t\#a \Rightarrow ua \in T \) follows from \( I/O\)-symmetry.

**If:** Assume \( T \) satisfies (8). We prove by induction on \( \ell(t) + \ell(u) \) that all states \( (t, u) \in rT \) are safe.

**Base:** \( \ell(t) + \ell(u) = 0 \), hence, \( t = \epsilon = u \). The state \( (\epsilon, \epsilon) \) is safe because \( T \) is non-empty and prefix-closed.

**Step:** \( \ell(t) + \ell(u) > 0 \), hence, we can find state \( (t', u') \in rT \) such that

\[ (t', u') \xrightarrow{T} (t, u). \]  \hspace{1cm} (9)

On account of the induction hypothesis, using that \( \ell(t') + \ell(u') < \ell(t) + \ell(u) \), we know that \( (t', u') \) is safe and, hence,

\[ t' \in T \land u' \in T. \]  \hspace{1cm} (10)

We distinguish two cases: \( t' = t \) and \( u' = u \). Because of \( I/O\)-symmetry we need only investigate the case \( t' = t \). In that case, \( u \) can be written as \( u'a \) for some symbol \( a \). Furthermore, \( t \in T \) follows from (10). All that we need to show now is \( u'a \in T \). We derive for the cases \( a \in I \) and \( a \in O \), respectively:

\[ a \in I \]
\[ \Rightarrow \{ (9), u = u'a, \text{ and definition of } \xrightarrow{T} \} \]
\[ u'a \in T \]

and

\[ a \in O \]
\[ \Rightarrow \{ (9), u = u'a, \text{ and definition of } \xrightarrow{T}, \text{ using } t = t' \} \]
\[ u'a \in T \]
\[ u\#a < t\#a \]
\[ \Rightarrow \{ (8), \text{ using } (t', u') \in rT \} \]
\[ u'a \in T \]

\[ \square \]

The preceding characterization can be simplified a little by introducing the *composability*\(^2\) relation \( C \) on \( A^* \) defined as \( r(A^*) \). That is, \( t \ C \ u \) holds when \( (t, u) \) is reachable under the specification \( A^* \). Specification \( A^* \) does not restrict transmission transitions and, thus, relation \( C \) captures only the restriction imposed by the condition that symbols arrive no earlier than they were sent.

\(^2\)The name 'composability' is taken from [4].
\textbf{Example 3} Assuming \(a \in I\) and \(b \in O\), we have \(\varepsilon \sqsubseteq a\) but \(\neg (a \sqsubseteq \varepsilon)\), and also \(b \sqsubseteq ab\) but \(\neg (ab \sqsubseteq ba)\). \(\square\)

Relation \(C\) enjoys a number of nice properties.

\textbf{Property 1} For specification \(T\) we have
\[
    r_T \cap (T \times T) = C \cap (T \times T).
\]
Furthermore, we have
\[
    a \in I \land \tau_a C u \equiv t \# a < u \# a \land t \sqsubseteq u
\]
\[\text{(12)}\]
\[
    a \in I \land t \sqsubseteq u \Rightarrow t C u a
\]
\[\text{(13)}\]
\[
    t C u a \Rightarrow (\exists t_0, t_1 : t = t_0 t_1 : t_0 C u)
\]
\[\text{(14)}\]
and \(C\) is reflexive, i.e. for all \(t\) we have \(t C t\). \(\square\)

Of course, on account of \(I/O\)-symmetry we also have dual forms of (12) through (14) obtained by interchanging \(I\) and \(O\), and left- and right-hand arguments of \(C\). For example, the dual of (13) is: \(a \in O \land u \sqsubseteq t \Rightarrow u a C t\).

We now give a characterization of delay-insensitivity in which reachability under the specification in question has been traded for \(C\).

\textbf{Theorem 2} Specification \(T\) is DI if and only if
\[
    (\forall t, u, a : t \in T \land u \in T)
    \begin{align*}
    (a \in I \land \tau_a C u & \Rightarrow ta \in T) \land \\
    (a \in O \land t C u a & \Rightarrow ua \in T)
    \end{align*}
\]
\[\text{(15)}\]
\textbf{Proof} On account of Theorem 1 it is sufficient to prove the equivalence of (8) and (15). We derive
\[
    (\forall t, u, a : (t, u) \in r_T \cap (T \times T)
    \begin{align*}
    : (t \# a < u \# a & \Rightarrow ta \in T) \land \\
    (u \# a > t \# a & \Rightarrow ua \in T)
    \end{align*}
\] = \{ \text{Property 1(11)} \}
\]
\[
    (\forall t, u, a : (t, u) \in C \cap (T \times T)
    \begin{align*}
    : (t \# a < u \# a & \Rightarrow ta \in T) \land \\
    (u \# a > t \# a & \Rightarrow ua \in T)
    \end{align*}
\] = \{ \text{predicate and set calculus} \}
\]
\[
    (\forall t, u, a : (t, u) \in (T \times T)
    \begin{align*}
    : (t \# a < u \# a \land t \sqsubseteq u & \Rightarrow ta \in T) \land \\
    (u \# a > t \# a \land t \sqsubseteq u & \Rightarrow ua \in T)
    \end{align*}
\] = \{ \text{Property 1(12)} \}
\]
\[
    (\forall t, u, a : (t, u) \in (T \times T)
    \begin{align*}
    : (a \in I \land t \sqsubseteq u & \Rightarrow ta \in T) \land \\
    (a \in O \land t \sqsubseteq u a & \Rightarrow ua \in T)
    \end{align*}
\]
This characterization can be further simplified:

**Theorem 3** Specification $T$ is DI if and only if

\[(\forall t, u, z : t \in T \land u \in T : t \subset C z \land z \subset C u \Rightarrow z \in T)\]  

(16)

Specification $T$ is called convex when it satisfies (16).

**Proof** On account of Theorem 2 it is sufficient to prove the equivalence of (15) and (16).

**If:** Assuming $T$ is convex we derive (15). Let $t \in T$ and $u \in T$. We derive

\[
\begin{align*}
a &\in I \land ta \subset C u \\
\Rightarrow &\quad \{ \text{reflexivity of } \subset \text{ on account of Property 1} \} \\
a &\in I \land t \subset C t \land ta \subset C u \\
\Rightarrow &\quad \{ \text{Property 1(13)} \} \\
t \subset C ta \land ta \subset C u \\
\Rightarrow &\quad \{ (16) \text{ assumed, using } t \in T \text{ and } u \in T \}
\end{align*}
\]

$ta \in T$

The other conjunct follows $I/O$-symmetrically.

**Only if:** Assuming (15) we prove (16) by induction on the length of $z$.

**Base:** $z = \varepsilon$. Since $T$ is non-empty and prefix-closed we have $z = \varepsilon \in T$.

**Step:** $z = z'a$. Assuming $t \in T$ and $u \in T$ such that $t \subset C z$ and $z \subset C u$, we show $z'a \in T$. We distinguish the cases $a \in I$ and $a \in O$. Because of $I/O$-symmetry we consider only the first case. Therefore, assume $a \in I$. On account of Property 1(14), using $a \in I$ and $t \subset C z'a$, we can let $t'$ be a prefix of $t$ such that $t' \subset C z'$. We derive

true

\[
\begin{align*}
&\quad \{ \text{context so far} \} \\
&\quad t \in T \land u \in T \land t' \subset C z' \land a \in I \land z'a \subset C u \\
\Rightarrow &\quad \{ t' \text{ is a prefix of } t, T \text{ is prefix-closed, and Property 1(12)} \} \\
&\quad t' \in T \land u \in T \land t' \subset C z' \land z' \subset C u \land a \in I \land z'a \subset C u \\
\Rightarrow &\quad \{ \text{induction hypothesis, using } \ell(z') < \ell(z) \} \\
&\quad z' \in T \land u \in T \land a \in I \land z'a \subset C u \\
\Rightarrow &\quad \{ (15) \text{ assumed} \} \\
z'a \in T
\end{align*}
\]
In [8] a proof is given for 'the JTU-Rules imply convexity'. That proof is based on a construction in terms of graphs. It is easy to understand if one is willing to accept some intuitions about graphs. A complete formalization is still quite lengthy. We will not take that road here. We postpone this implication and instead concentrate on the (easier) converse.

Before we tackle the converse it is useful to get to know the composability relation $C$ a little better. We call $(t'a, u'b)$ an inversion in $(t, u)$ when

- $t'a$ is a prefix of $t$,
- $u'b$ is a prefix of $u$,
- $t'a#a > u'b#a$, and
- $t'a#b < u'b#b$.

The first condition expresses that $t'a$ locates an occurrence of symbol $a$ in trace $t$ and, similarly, $u'b$ locates an occurrence of $b$ in $u$ on account of the second condition. The third condition expresses that the occurrence of $a$ in $u$, that corresponds to $t'a$, occurs to the 'right' of $u'b$—if it exists at all. Similarly, the fourth condition expresses that the occurrence of $b$ in $t$ which corresponds to $u'b$ occurs to the 'right' of $t'a$. Hence, the order of these occurrences of $a$ and $b$ in $t$ differs from the order of the corresponding occurrences in $u$. The set of inversions in state $(t, u)$ will be denoted by $\text{inv}(t, u)$. Inversion $(t'a, u'b)$ in $(t, u)$ is called a $t$-neighbor inversion when $t'ab$ is a prefix of $t$ and $t'a#b = u'#b$, that is, when these occurrences of $a$ and $b$ are adjacent in $t$.

Notice that the concept of inversion does not involve the directions of symbols, i.e., it is independent of how $A$ is partitioned into $I$ and $O$. We will only be interested in inversions in states $(t, u)$ for which $t# = u#$. Let us look at an example.

**Example 4** Assuming that symbols $a$, $b$, and $c$ are distinct, there are three inversions

![Figure 0: Inversion diagram for (abca, caab)](image)

in $(abca, caab)$. The set $\text{inv}(abca, caab)$ consists of $(a, c)$, $(ab, c)$, and $(ab, caa)$. Only one of these, viz. $(ab, c)$, is an $abca$-neighbor inversion. This is illustrated in Figure 0, where line segments connect corresponding symbol occurrences. Each pair of intersecting line segments corresponds to an inversion.

We draw the attention to some well-known properties of inversions to be used later on:
Property 2  For state \((t,u)\) such that \(t# = u#\) we have
\[
\text{inv}(t,u) \text{ is finite, }
\text{inv}(t,u) = \emptyset \iff t = u,
\text{inv}(t,u) \neq \emptyset \iff \text{inv}(t,u) \text{ contains a } t\text{-neighbor inversion.}
\]
If \((ta,u'b)\) is a \(tabv\)-neighbor inversion in \((tabv,u)\), then
\[
\text{inv}(tabv,u) = \text{inv}(tabv,u) - \{(ta,u'b)\}.
\]

Composability can be characterized in terms of inversions:

Property 3  For state \((t,u)\) we have
\[
t \subseteq u \Rightarrow (\forall t',u',a,b: (t'a,u'b) \in \text{inv}(t,u) : a \in O \lor b \in I)
\]
and this is an equivalence if \(t# = u#\).

Example 5  Assuming \(a \in I\) and \(b \in O\), we have seen in Example 3 that \(\neg (ab \subseteq ba)\) holds. State \((ab, ba)\) has one inversion, viz. \((a,b)\), for which \(\neg (a \in O \lor b \in I)\).

By the way, from Property 3 it follows (non-trivially) that \(\subseteq\) is transitive (hence, a pre-order), but we will not need that here.

We are now ready for

Theorem 4  If specification \(T\) is convex (cf. (16)), then it satisfies the JTU-Rules.

Proof  Assume specification \(T\) is convex. Each JTU-Rule can be viewed as a special case of convexity. We prove them one by one.

Rule \(R_3\): For traces \(s\) and \(t\), and symbols \(a\) and \(b\) of the same direction we have, on account of Property 3,
\[
sabt \subseteq sbat \subseteq sabt.
\]
Using convexity we now infer \(sabt \in T \Leftrightarrow sbat \in T\) and, hence, \(T\) satisfies \(R_3\).

Rule \(R'_3\): For traces \(s\) and \(t\), and symbols \(a\), \(b\), and \(c\) such that the direction of \(a\) and \(c\) differs from that of \(b\), we have, using Properties 3 and 1(13),
\[
a \in O \Rightarrow sabtc \subseteq sbatc \subseteq sabtc
\]
On account of convexity we then have that \(sabtc \in T \land sbat \in T\) implies \(sbatc \in T\) and, hence, \(T\) satisfies \(R'_3\).

Rule \(R'_5\): For trace \(s\) and symbols \(a\) and \(b\) of different direction we have, using reflexivity of \(\subseteq\) and Properties 1(12) and 1(13),
\[
a \in O \Rightarrow sa \subseteq sab \subseteq sb
\]
On account of convexity we then have that \(sa \in T \land sb \in T\) implies \(sab \in T\) and, hence, \(T\) satisfies \(R'_5\).
4 New Representation for Specifications

The major innovation in our proof of the Fundamental Characterization Theorem of Delay-Insensitivity is a new representation for specifications. This representation is based on enhanced characteristic functions, or ECFs for short. An ECF is a mapping from $A^*$ to $\{0, 1, 2\}$. The enhancement consists of the additional value 2 in the co-domain, which enables us to distinguish two ways in which a trace does not belong to a specification. We use $\cdot$ to denote functional application for ECFs, which has a weaker binding power than concatenation, i.e. $f \cdot st$ stands for $f(st)$.

Let $T$ be a specification. Observe that each trace $t \notin T$ can be uniquely written as $t_0at_1$ such that $t_0 \in T$ and $t_0a \notin T$, since $T$ is non-empty and prefix-closed. We now define ECF $f_T$ by

$$f_T : t = \begin{cases} 0 & \text{if } (\exists t_0,a,t_1 : t = t_0at_1 \land t_0 \in T \land t_0a \notin T : a \in O) \\ 1 & \text{if } t \in T \\ 2 & \text{if } (\exists t_0,a,t_1 : t = t_0at_1 \land t_0 \in T \land t_0a \notin T : a \in I) \end{cases}$$

Let $f = f_T$, then $f$ enjoys the following properties:

- $(F_0)$ $f \varepsilon = 1$
- $(F_1)$ $f \cdot ta = f \cdot t$ if $f \cdot t \neq 1$
- $(F_2)$ $f \cdot ta \leq f \cdot t$ if $a \in O$
- $(F_3)$ $f \cdot ta \geq f \cdot t$ if $a \in I$

These follow immediately from the definition of $f_T$ and the fact that $T$ is non-empty and prefix-closed. Properties $F_1$ through $F_3$ are readily generalized to

- $(F_1')$ $f \cdot tu = f \cdot t$ if $f \cdot t \neq 1$
- $(F_2')$ $f \cdot tu \leq f \cdot t$ if $u \in O^*$
- $(F_3')$ $f \cdot tu \geq f \cdot t$ if $u \in I^*$

by induction on the length of $u$. Furthermore, $F_1 \land F_2 \land F_3$ is equivalent to the conjunction of

- $(F_2'')$ $f \cdot ta < f \cdot t \Rightarrow f \cdot t = 1 \land f \cdot ta = 0 \land a \in O$
- $(F_3'')$ $f \cdot ta > f \cdot t \Rightarrow f \cdot t = 1 \land f \cdot ta = 2 \land a \in I$

For ECF $f$ we define its trace set $tf$ by

$$tf = \{ t \mid f \cdot t = 1 \}.$$  \hfill (19)

We now trivially have for $T \subseteq A^*$

$$t(fT) = T.$$  \hfill (20)

We also claim that for ECF $f$ satisfying $F_0$ through $F_3$ we have that $tf$ is non-empty and prefix-closed (viz. on account of $F_0$ and $F_1$) and that
\[ f(t_f) = f. \quad (21) \]

We have thus established a one-one correspondence between specifications and ECFs satisfying \( F_0 \) through \( F_3 \). Notice that exchanging the role of \( I \) and \( O \) corresponds to exchanging the role of 0 and 2 in ECFs.

Using the ECF of a specification, the JTU-Rules can be condensed into a single rule. We prove only an implication here; the converse will follow from Theorems 6, 7, and 4.

**Theorem 5** If specification \( T \) satisfies the JTU-Rules then

\[(\forall s, a, b, t : a \in O \lor b \in I : fT \cdot sabt \leq fT \cdot sbat).\]  

(22)

Predicate (22) will be called the *Neighbor-Swap Rule.*

**Proof** Assuming \( T \) satisfies the JTU-Rules we show that \( T \) satisfies the Neighbor-Swap Rule. Because the co-domain of \( fT \) is \( \{0, 1, 2\} \) and

\[
\begin{array}{ccc}
0 & 1 & 2 \\
0 & \leq & \leq \\
1 & > & \leq \\
2 & > & \leq \\
\end{array}
\]

it is sufficient to prove for all traces \( s \) and \( t \), and symbols \( a \) and \( b \) such that \( a \in O \lor b \in I \):

\[
\begin{align*}
fT \cdot sabt = 2 & \Rightarrow fT \cdot sbat = 2 \quad \text{and} \\
fT \cdot sbat = 0 & \Rightarrow fT \cdot sabt = 0. 
\end{align*}
\]

On account of \( I/O \)-symmetry we confine ourselves to the first of these. Therefore, also assume that \( fT \cdot sabt = 2 \). From the definition of \( fT \) now follows that we can find \( u \) and \( c \) such that

\[
u c \text{ prefix of } sabt \land u \in T \land uc \notin T \land c \in I. \quad (23)
\]

Our goal is to show that \( fT \cdot sbat = 2 \) as well. We distinguish four cases: \( uc \) prefix of \( s \), \( uc = sa \), \( uc = sab \), and \( sab \) prefix of \( u \).

**Case \( uc \) prefix of \( s \):** Then \( fT \cdot sbat = 2 \) by (23) and the definition of \( fT \).

**Case \( uc = sa \):** We derive

\[
\begin{align*}
uc &= sa \\
= & \{ \text{trace calculus} \} \\
u &= s \land s = a \\
\Rightarrow & \{ (23) \} \\
s & \in T \land sa \notin T \land a \in I \\
\Rightarrow & \{ T \text{ is prefix-closed and } a \in O \lor b \in I \text{ assumed} \} \\
s & \in T \land sab \notin T \land a \in I \land b \in I
\end{align*}
\]

12
\[
= \quad \{ \text{Rule } R_3 \text{ assumed } \}
\]
\[
s \in T \land sba \notin T \land a \in I \land b \in I
\]
\[
\Rightarrow \quad \{ \text{predicate calculus, doing case analysis on } sb \in T \}
\]
\[
(s \in T \land sb \notin T \land b \in I) \lor (sb \in T \land sba \notin T \land a \in I)
\]
\[
\Rightarrow \quad \{ \text{definition of } fT \}
\]
\[
fT \cdot sbat = 2
\]

**Case uc = sab**: We derive

\[
uc = sab
\]
\[
= \quad \{ \text{trace calculus } \}
\]
\[
u = sa \land c = b
\]
\[
\Rightarrow \quad \{ (23) \}
\]
\[
sa \in T \land sba \notin T \land b \in I
\]
\[
\Rightarrow \quad \{ \text{predicate calculus } \}
\]
\[
(sa \in T \land sba \notin T \land a \in O \land b \in I) \lor (sa \in T \land sab \notin T \land a \in I \land b \in I)
\]
\[
\Rightarrow \quad \{ \text{Rules } R_3^{''} \text{ and } R_3 \text{ assumed and } T \text{ is prefix-closed } \}
\]
\[
(s \in T \land sb \notin T \land b \in I) \lor (s \in T \land sba \notin T \land a \in I \land b \in I)
\]
\[
\Rightarrow \quad \{ \text{predicate calculus, doing case analysis on } sb \in T \}
\]
\[
(s \in T \land sb \notin T \land b \in I) \lor (sb \in T \land sba \notin T \land a \in I)
\]
\[
\Rightarrow \quad \{ \text{definition of } fT \}
\]
\[
fT \cdot sbat = 2
\]

**Case sab prefix of u**: Hence, we can write \( t = t_0c t_1 \) such that \( u = sbt_0 \). We distinguish two subcases depending on the equality of the directions of \( a \) and \( b \). For \( a \) and \( b \) having the same direction we derive

\[
u = sbt_0
\]
\[
\Rightarrow \quad \{ (23) \}
\]
\[
sbt_0 \in T \land sbt_0c \notin T \land c \in I
\]
\[
= \quad \{ \text{Rule } R_3 \text{ assumed, using that } a \text{ and } b \text{ have same direction } \}
\]
\[
 sbt_0 \in T \land sbt_0c \notin T \land c \in I
\]
\[
\Rightarrow \quad \{ \text{definition of } fT \}
\]
\[
fT \cdot sbat = 2
\]

For symbols \( a \) and \( b \) with different directions we proceed as follows. From the assumption \( a \in O \lor b \in I \) we now infer \( a \in O \land b \in I \) and we derive

\[
u = sbt_0
\]
\[
\Rightarrow \quad \{ (23) \}
\]

13
sabt₀ ∈ T ∧ sabt₀c ∉ T ∧ c ∈ I
⇒ \{ Rule R'₄ assumed, using that a ∈ O ∧ b ∈ I \} sabt₀ ∈ T ∧ sabt₀c ∉ T

Hence, we can find prefix vd of sabt₀c such that v ∈ T but vd ∉ T. On account of the definition of fT it is sufficient to show d ∈ I in order to have fT·sbat = 2. Finally, we distinguish the five ways in which vd can be a prefix of sabt₀c, viz. vd prefix of s, vd = sb, vd = sba, vd prefix of sabt₀ with sba prefix of v, and vd = sabt₀c. The first case is excluded by s ∈ T. In the second case we have d = b ∈ I. The third case is excluded by sa ∈ T and Rule R''₅. For the fourth case note that d ∈ O is excluded by Rule R''₄ and the fact that sabt₀d is a prefix of sabt₀ with sabt₀ ∈ T. In the last case we have d = c ∈ I. This completes the proof.

The Neighbor-Swap Rule can be generalized as follows:

**Theorem 6** Specification T satisfies the Neighbor-Swap Rule (22) if and only if

\[(∀ t, u : t C u : fT·t ≤ fT·u). \quad (24)\]

We say that specification T is **monotonic** if it satisfies (24).

**Proof**

**If:** That (24) implies (22) follows immediately from

\[a ∈ O ∨ b ∈ I ⇒ sabt C sbat,\]

which is a consequence of Property 3.

**Only if:** Assuming T satisfies the Neighbor-Swap Rule we prove that T is monotonic. Let t and u be such that t C u. We first deal with the case where t# = u#. We prove fT·t ≤ fT·u by induction on the number of inversions in (t, u).

**Base:** inv(t, u) = 0. Hence, on account of Property 2 using t# = u#, we have t = u and, thus, fT·t ≤ fT·u.

**Step:** inv(t, u) ≠ 0. Hence, on account of Property 2 using t# = u#, there exists a t-neighbor inversion in (t, u), say, (t₀a, u₀b). Therefore, we can write t = t₀abt₁. From Property 3 and assumption t C u follows a ∈ O ∨ b ∈ I. We now derive

\[fT·t = \{ t = t₀abt₁ \} \]
\[fT·t₀abt₁ \leq \{ T satisfies the Neighbor-Swap Rule by assumption, using a ∈ O ∨ b ∈ I \} \]
\[fT·t₀bat₁ \leq \{ induction hypothesis, using inv(t₀bat₁, u) ⊆ inv(t₀abt₁, u) by Property 2 \} \]
\[fT·u]
Finally, we consider the other case where \( t\# \neq u\#. \) In that case we can find \( v \in I^* \) and \( w \in O^* \) such that \( tv\# = uw\# \) and \( tv \preceq uw \) on account of Property 1(12). We now derive

\[
\begin{align*}
& fT \cdot t \\
& \leq \quad \{ (F^\#_v), \text{using } v \in I^* \} \\
& fT \cdot tv \\
& \leq \quad \{ \text{first case, using } tv\# = uw\# \text{ and } tv \preceq uw \} \\
& fT \cdot uw \\
& \leq \quad \{ (F^\#_w), \text{using } w \in O^* \} \\
& fT \cdot u
\end{align*}
\]

At last, we can close the gap between delay-insensitivity and the JTU-Rules:

**Theorem 7** If specification \( T \) is monotonic (cf. (24)) then it is convex (cf. (16)).

**Proof** Assuming \( T \) is monotonic, we show that it is convex. Let \( t \in T \) and \( u \in T \) such that \( t \preceq z \) and \( z \preceq u \). We derive

\[
\begin{align*}
& 1 \\
= & \quad \{ \text{definition of } fT, \text{using } t \in T \} \\
& fT \cdot t \\
\leq & \quad \{ \text{monotonicity assumed, using } t \preceq z \} \\
& fT \cdot z \\
\leq & \quad \{ \text{monotonicity assumed, using } z \preceq u \} \\
& fT \cdot u \\
= & \quad \{ \text{definition of } fT, \text{using } u \in T \} \\
& 1 \\
\end{align*}
\]

Hence, \( fT \cdot z = 1 \) and from the definition of \( fT \) now follows \( z \in T \).

We conclude this section with the proof for Theorem 0:

**Proof** On account of Theorem 3 it is sufficient to show that convexity is equivalent to the JTU-Rules. We show the two implications in one derivation:

\[
\begin{align*}
& T \text{ is convex} \\
\Rightarrow & \quad \{ \text{Theorem 4} \} \\
& T \text{ satisfies the JTU-Rules} \\
\Rightarrow & \quad \{ \text{Theorem 5} \} \\
& T \text{ satisfies the Neighbor-Swap Rule} \\
= & \quad \{ \text{Theorem 6} \} \\
& T \text{ is monotonic} \\
\Rightarrow & \quad \{ \text{Theorem 7} \} \\
& T \text{ is convex}
\end{align*}
\]
5 Concluding Remarks

We have studied protocols for asynchronous communication between two parties over an interface of directed channels. Non-empty prefix-closed trace sets have been used to specify communication protocols. Such a specification embodies restrictions on the initiation of transmissions and the readiness for receptions, for both parties. An operational semantics for the communication activity has been given in terms of a transition system. We have defined the notion of a delay-insensitive protocol specification based on absence of computation interference as a correctness concern. This correctness concern derives from an interpretation of the model as an abstraction of digital circuit physics.

The central problem of this paper has been the characterization of delay-insensitive protocol specifications. In summary, we have shown that for all protocol specifications \( T \) the following statements are equivalent:

- \( T \) is delay-insensitive (DI)
- \( T \) satisfies (8)
- \( T \) satisfies (15)
- \( T \) is convex (cf. (16))
- \( T \) is monotonic (cf. (24))
- \( T \) satisfies the Neighbor-Swap Rule (cf. (22))
- \( T \) satisfies the JTU-Rules

The characterization with JTU-Rules is due to [4] and that with convexity first appears in [7]. The characterizations in terms of monotonicity and the Neighbor-Swap Rule are new. Both are based on a new representation of protocol specifications by means of enhanced characteristic functions. The Neighbor-Swap Rule and monotonicity have turned out to be convenient stepping stones for a new proof of the Fundamental Characterization Theorem of Delay-Insensitivity.

Because of its simplicity, the Neighbor-Swap Rule is preferable to the JTU-Rules, for example, when checking a specification for delay-insensitivity. We should point out, however, that Udding [4] used variations on the JTU-Rules to classify delay-insensitive specifications. This classification is not obvious in terms of the Neighbor-Swap Rule and the variations are also easier to check in the minimal-deterministic-state-graph representation of specifications.

Dill's canonical process descriptions in [1] can be related to ours as follows. Protocol specification \( T \) has canonical process description

\[
\langle I, O, T, \{t \mid fT\cdot t = 2\} \rangle
\]
and canonical process description $\langle I, O, S, F \rangle$ corresponds to protocol specification $S$ (recall that the sets $I$ and $O$ are fixed in our context; the $F$-component of a canonical process description is superfluous). Our new representation in terms of the enhanced characteristic function is so nice because it maintains the $I/O$-symmetry and, thus, allows a uniform treatment of the three sets $S$, $F$, and $F = (I \cup O)^* - (S \cup F)$.

The partial order $\subseteq$ on specifications defined in [6] corresponds to the point-wise order on enhanced characteristic functions. For specifications $S$ and $T$ we have

$$S \subseteq T \equiv (\forall s :: fS \cdot s \leq fT \cdot s).$$

This property greatly simplifies the analysis of the $\subseteq$-lattice of protocol specifications. The alternative representation $T'$ of specifications suggested in [6] consists of pairs

$$\{\{t | fT \cdot t \leq 1\}, \{t | fT \cdot t = 2\}\}.$$  Both these sets are C-upward closed for DI specifications. The relation nai of [6] enjoys the property

$$S \text{ nai } T \equiv (\forall s,t :: s \subseteq t : fS \cdot s \leq fT \cdot t)$$

and, therefore, the ECF $\hat{f}$ of $T$'s DI-equivalent, i.e. of lub.$[T]$, satisfies

$$\hat{f} = (\text{MAX } s :: s \subseteq t : fT \cdot s).$$

In this paper we have dealt with the case of two parties communicating according to a single protocol specification. In [6] general networks of asynchronously communicating processes are studied. There, it is also shown that the special case of a closed network consisting of two processes with the same trace set plays an important role in defining a denotational semantics.

The relationship with [2] by Josephs et al. is also prominent. Their relation $\subseteq$ on traces can be expressed as follows:

$$u \subseteq t \equiv t \subseteq u \land t\# = u\#.$$  They denote an asynchronous process by a pair $\langle F, D \rangle$ of trace sets satisfying certain closure properties. Because of these closure properties, the trace sets $F$ and $D$ can be reconstructed from $F - D$. The prefix-closures of these difference sets, i.e. $\hat{F} - D$, precisely span our space of DI specifications.

In this paper we have investigated safety aspects only. Liveness aspects can be incorporated, but this requires a more refined notion of protocol specification and, in general, a more subtle way of defining the operational semantics. This will be reported on in a separate paper. It results in a specification space isomorphic to the one presented in [2].
References


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>
<tr>
<td>Year</td>
<td>Author(s)</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>86/14</td>
<td>R. Koymans</td>
<td>Specifying passing systems requires extending temporal logic</td>
</tr>
<tr>
<td>87/01</td>
<td>R. Gerth</td>
<td>On the existence of sound and complete axiomatizations of the monitor concept</td>
</tr>
<tr>
<td>87/02</td>
<td>Simon J. Klaver, Chris F.M. Verberne</td>
<td>Federatieve Databases</td>
</tr>
<tr>
<td>87/03</td>
<td>G.J. Houben, J. Paredaens</td>
<td>A formal approach to distributed information systems</td>
</tr>
<tr>
<td>87/04</td>
<td>T. Verhoeff</td>
<td>Delay-insensitive codes - An overview</td>
</tr>
<tr>
<td>87/05</td>
<td>R. Kuiper</td>
<td>Enforcing non-determinism via linear time temporal logic specification.</td>
</tr>
<tr>
<td>87/06</td>
<td>R. Koymans</td>
<td>Temporele logica specificatie van message passing en real-time systemen (in Dutch).</td>
</tr>
<tr>
<td>87/07</td>
<td>R. Koymans</td>
<td>Specifying message passing and real-time systems with real-time temporal logic.</td>
</tr>
<tr>
<td>87/08</td>
<td>H.M.J.L. Schols</td>
<td>The maximum number of states after projection.</td>
</tr>
<tr>
<td>87/10</td>
<td>T. Verhoeff</td>
<td>Three families of maximally nondeterministic automata.</td>
</tr>
<tr>
<td>87/11</td>
<td>P. Lemmens</td>
<td>Eldorado ins and outs. Specifications of a data base management toolkit according to the functional model.</td>
</tr>
<tr>
<td>87/12</td>
<td>K.M. van Hee, A. Lapinski</td>
<td>OR and AI approaches to decision support systems.</td>
</tr>
<tr>
<td>87/13</td>
<td>J.C.S.P. van der Woude</td>
<td>Playing with patterns, searching for strings.</td>
</tr>
<tr>
<td>87/14</td>
<td>J. Hooman</td>
<td>A compositional proof system for an occam-like real-time language</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------</td>
</tr>
<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>
<tr>
<td>Year</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>88/10</td>
<td>J.C. Ebergen</td>
<td>A Formal Approach to Designing Delay Insensitive Circuits</td>
</tr>
<tr>
<td>88/11</td>
<td>G.J. Houben, J.Paredaens</td>
<td>A graphical interface formalism: specifying nested relational databases</td>
</tr>
<tr>
<td>88/12</td>
<td>A.E. Eiben</td>
<td>Abstract theory of planning</td>
</tr>
<tr>
<td>88/13</td>
<td>A. Bijlsma</td>
<td>A unified approach to sequences, bags, and trees</td>
</tr>
<tr>
<td>88/14</td>
<td>H.M.M. ten Eikelder, R.H. Mak</td>
<td>Language theory of a lambda-calculus with recursive types</td>
</tr>
<tr>
<td>88/15</td>
<td>R. Bos, C. Hemerik</td>
<td>An introduction to the category theoretic solution of recursive domain equations</td>
</tr>
<tr>
<td>88/16</td>
<td>C. Hemerik, J.P. Katoen</td>
<td>Bottom-up tree acceptors</td>
</tr>
<tr>
<td>88/17</td>
<td>K.M. van Hee, G.J. Houben, L.J. Somers, M. Voorhoeve</td>
<td>Executable specifications for discrete event systems</td>
</tr>
<tr>
<td>88/18</td>
<td>K.M. van Hee, P.M.P. Rambags</td>
<td>Discrete event systems: concepts and basic results.</td>
</tr>
<tr>
<td>88/19</td>
<td>D.K. Hammer, K.M. van Hee</td>
<td>Fasering en documentatie in software engineering.</td>
</tr>
<tr>
<td>88/20</td>
<td>K.M. van Hee, L. Somers, M. Voorhoeve</td>
<td>EXSPECT, the functional part.</td>
</tr>
<tr>
<td>89/1</td>
<td>E.Zs.Lepoeter-Molnar</td>
<td>Reconstruction of a 3-D surface from its normal vectors.</td>
</tr>
<tr>
<td>89/2</td>
<td>R.H. Mak, P.Struik</td>
<td>A systolic design for dynamic programming.</td>
</tr>
<tr>
<td>89/3</td>
<td>H.M.M. Ten Eikelder, C. Hemerik</td>
<td>Some category theoretical properties related to a model for a polymorphic lambda-calculus.</td>
</tr>
<tr>
<td>89/4</td>
<td>J.Zwiers, W.P. de Roever</td>
<td>Compositionality and modularity in process specification and design: A trace-state based approach.</td>
</tr>
<tr>
<td>89/5</td>
<td>Wei Chen, T.Verhoeff, J.T. Udding</td>
<td>Networks of Communicating Processes and their (De-)Composition.</td>
</tr>
<tr>
<td>89/6</td>
<td>T.Verhoeff</td>
<td>Characterizations of Delay-Insensitive Communication Protocols.</td>
</tr>
<tr>
<td>89/7</td>
<td>P.Struik</td>
<td>A systematic design of a paralell program for Dirichlet convolution.</td>
</tr>
</tbody>
</table>