Delay-insensitive communication

Delay-insensitive Communication

Huub Schols

Delay-insensitive Communication

PROEFSCHRIFT

TER VERKRIJGING VAN DE GRAAD VAN DOCTOR AAN DE TECHNISCHE UNIVERSITEIT EINDHOVEN, OP GEZAG VAN DE RECTOR MAGNIFICUS, PROF. DR. J.H. VAN LINT, VOOR EEN COMMISSIE AANGEWEZEN DOOR HET COLLEGE VAN DEKANEN IN HET OPENBAAR TE VERDEDIGEN OP WOENSDAG 9 DECEMBER 1992 OM 14.00 UUR

DOOR

Hubert Marie Jean Louis Schols

GEBOREN TE AMBY

Dit proefschrift is goedgekeurd door de promotoren

prof. dr. M. Rem

en

prof. C.E. Molnar, Sc. D.

to my friends

If the car industry behaved like the computer industry over the last 30 years,
a Rolls-Royce would cost $5, get 300 miles per gallon,
and blow up once a year killing all passengers inside.

origin unknown

Acknowledgments

The author thanks the members of the Institute for Biomedical Computing of Washington University in St.Louis for their contributions to his understanding of the material in this monograph; in particular the intense and enthusiastic co-operation of Charles E. Molnar has sharpened many ideas and has been very stimulating during the development of the theory presented. 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, Wilbert Körver, Tom Verhoeff, Frans Kruseman Aretz, and Cees Jan Koomen. Thanks also go to the members of the Department of Computer Science of Washington University, particularly Jerome R. Cox jr. and Takayuki (Dan) Kimura, to Alain Martin and his graduate students for their helpful criticisms and suggestions at several presentations of parts of this material, to Mohammed Gouda, and to the members of the Eindhoven VLSI club for their criticisms on this material. Furthermore, the author thanks Occo Nolf for pointing out many typographical errors in the final draft of this monograph.

Contents

0

Introduction

In the middle of the 20th century Huffman, cf. [Huffman54], and Muller and Bartky, cf. [Muller-Bartky59], started to develop theories for designing asynchronous circuits. Since then, interest in asynchronous design has existed at just a few places. Only in the last decade, asynchronous design seems to have become a topic of general interest, cf. [Barney85], culminating in Sutherland's Turing Award lecture, see [Sutherland89], and spreading over many research institutes since then.

0.0 Synchronous and asynchronous

In this section we indicate how we interpret the terms "synchronous" and "asynchronous". These interpretations are inspired by [Molnar92].

The terms "synchronous" and "asynchronous" have been used with different meanings in different contexts. As applied to circuits, the terms have generally distinguished those

> that employ a "clock signal" that serves as a reference to separate consecutive circuit states from one another

from those

> that do not make use of such a signal, but that define states in terms of input values and internal actions that result in changes of circuit conditions.

Some circuits, such as those designed with "fundamental mode" restrictions on the changes of input values, may be interpreted either way.

As applied to communications rather than circuits, the term "synchronous" has been used to mean that the sending and the reception of a communication signal are regarded as the same event. In the case of CSP, cf. [Hoare85], there is an even stronger requirement that both "sender" and "receiver" must agree upon a communication signal, and hence that there is no distinction between sender and receiver. In a more general case, the term "synchronous" has been taken to mean that there is no delay between the sending of a signal and its reception, or, more abstractly, that the actions of sending and receiving a particular signal each stand in precisely the same ordering relation to other signaling actions. In comparison, "asynchronous" communication signals have distinct sending and receiving actions associated with them, which, in general, have different ordering relations to other signaling actions. In other words, there may be a non-zero delay between the sending of an asynchronous communication signal and its arrival at the receiver.

The different usages of the terms "synchronous" and "asynchronous" have arisen in the context of, on the one hand, circuit design and, on the other hand, abstract process communication models. They threaten no confusion when used exclusively within these distinct domains. Opportunities for severe confusion arise when these domains overlap, as they do in the discussion of the design of circuits to implement structures that are defined in the language and formalism of communication models, as in this monograph.

At this point we want to distinguish and discuss three kinds of communication:

(i) The sending of a communication signal and its arrival are not identified; there is a condition that the arrival of such a signal must not precede the sending of this signal. The sender alone controls if and when a signal is sent.

(ii) The sending of a communication signal and its arrival are identified; the sending and arrival actions of each communication signal are identically ordered with respect to all other actions. The sender alone controls if and when a signal is sent.

(iii) The sending of a communication signal and its arrival are identified; the sending and arrival actions of each communication signal are identically ordered with respect to all other actions. Both sender and receiver jointly control if and when such a joint action occurs. As a consequence, there is no difference in the role of "sender" and "receiver".

There may exist general agreement that (i) and (iii) are in the categories "asynchronous communication" and "synchronous communication", respectively; however, (ii) might be classified either way. In this monograph we discuss communication in a physical context. We consider (ii) to be in the

category "synchronous communication"; the connection between components that model mechanisms that communicate as described by (ii) is called *direct*. Furthermore, the connection between components that model mechanisms that communicate as described by (i) is called *indirect*. In the kind of communication described by (iii) sender and receiver share the control whether and, if so, when a joint action occurs; we consider this to constitute a higher level communication primitive that falls outside the scope of this monograph.

0.0.0 Asynchronous communication

There exist various reasons why one may be interested in asynchronous communication. Here, we mention scaling, variable or unknown delays, and metastability.

When integrating circuits at an increasingly larger scale, delays in the interconnections between the switching elements tend to increase relatively to the delays in the switching elements, cf. [Seitz80, van de Snepscheut85]. In order to obtain a lot of freedom for placement and routing, we are interested in separating the functional and geometrical design tasks. This can be established by designing circuits that behave correctly independent of the size of the delays. This goal is achieved in the area "delay-insensitive communication" in the discipline "asynchronous communication design".

Another source of motivation for studying delay-insensitive communication is the occurrence of metastable behavior in digital circuits. We consider a system that has a continuous state space with at least two stable states and at least one unstable state. The system will converge to one of its stable states. Which stable state the system will end up in depends on the initial condition. For such a system and a given finite interval of time, there exists an initial condition such that the system doesn't reach any stable state within this interval. This phenomenon is called *metastability*. Chaney and Molnar, cf. [Chaney – Molnar73], presented experimental evidence showing metastable behavior in digital circuits. Hurtado, cf. [Hurtado75], argued that metastable behavior is an important and intrinsic issue; therefore we mention it next to (other) variable delays, see also [Kleemann – Cantoni87].

Furthermore, asynchronous communication can be used as a model for the communication in distributed systems, e.g. transputers, Cosmic Cubes, cf. [Seitz85, Dally – Seitz86], or the FFP-machine, cf. [Mago85]. Asynchronous communication can also be used in an interface between internally synchronous parts.

In this monograph we address communication between mechanisms. Mechanisms communicate by sending and receiving (physical) signals. We treat communication between mechanisms that are modeled to have an indirect connection. This results in the formal definitions of delay-safe and delay-insensitive communication. Our notion of delay-safe (and also delay-insensitive) communication comprises that the value of the delay between the sending and the reception of each such signal has an unknown non-negative value.

0.0.1 Communication Model

We introduce the formal Communication Model. In the Communication Model we use trace theory as a tool. The trace theory formalism has been developed at Eindhoven University of Technology by Rem and others, cf. [Rem85, Rem – van de Snepscheut – Udding83, van de Snepscheut85, Kaldewaij86]. The interpretation of trace theory in the Communication Model yields a formalization of delay-safe (and delay-insensitive) communication. Our research is concerned with three topics:

- delay-safe communication,
- delay-insensitive communication, and
- absence of computation interference hazard.

We address these topics at three levels:

- the relation between the Communication Model and the underlying physics,
- notions in the Communication Model and the relations between them, and
- the use of the trace theory formalism in the Communication Model.

Although we like to play formal games, the formal game presented in this monograph has been inspired by physical problems. We think that the material presented in this monograph may be a helpful tool for designers who are concerned with asynchronous communication; we show the limitations of delay-safe and delay-insensitive communication. Furthermore, our work provides a starting point for the integration of synchronous communication design and asynchronous communication design.

0.0.2 Computation interference hazard

Molnar and Fang pointed out that a specification of the mechanism to be designed should not only be interpreted as a specification for the mechanism itself, but that, in general, such a specification puts restrictions upon the communication between the mechanism and its environment, see [Molnar – Fang83]. Our study of

asynchronous communication has revealed the urge to distinguish between the reception (arrival) of a signal and its acceptance. The arrival of a signal at a moment that it cannot be accepted by a mechanism is called "computation interference". The danger that this might happen is called computation interference hazard. The correctness concern "absence of computation interference hazard" is the basic correctness concern in this monograph. The distinction between the "reception" and "acceptance" of a signal provides the context that is needed for the discussion of computation interference hazard. Our interest in the correctness concern "absence of computation interference hazard" originally has emerged within the context of "asynchronous communication". Separating the correctness concern from this context has enabled us to address synchronous as well as asynchronous communication, using direct and indirect connections respectively, within one formal framework: our Communication Model.

0.1 Subsequent chapters

In chapter 1 we present some tools that we use in this monograph. The Communication Model is presented in chapter 2. We use the word "model" to relate notions in our Communication Model to notions in the underlying physics. Our Communication Model provides a clear separation between the interpretation of physical issues and the formalism. We distinguish between the communication behavior of a module and the communication of an interconnection. Furthermore, we introduce abstractions: we define components as equivalence classes of modules and we define channels as equivalence classes of interconnections. We address computation interference hazard in chapter 3. Absence of computation interference hazard being our basic correctness concern, we present a technique to transform other correctness concerns into absence of computation interference hazard. In chapter 4 we are concerned with delay-safe communication; absence of computation interference hazard is the correctness concern. In this chapter we focus on the communication behavior of mechanisms that communicate in a delay-safe way. Within the context of delay-safe communication, we address in chapter 5 an additional correctness concern, viz. absence of transmission interference hazard. Transmission interference hazard models that it is possible that some signals interfere with each other. The communication is delay-insensitive if and only if the communication is delay-safe and there is no transmission interference hazard. In chapter 6 we address composition and decomposition. We present necessary and sufficient conditions for composition under some given correctness concerns and a method to calculate the composition under these conditions. In this chapter we are concerned with connections that are partially direct and partially indirect. Within our study at the

level of process communication, an indirect connection between components models allowing for delays of unknown size in signals exchanged between mechanisms, whereas a direct connection between components models allowing only for zero delays in signals exchanged between mechanisms. Both direct and indirect connections are discussed in chapter 6. We present a relation between our research and the work of others in chapter 7; there, we also give some concluding remarks and we pinpoint some topics for future research.

0.2 Denotations in the English language

We use double quotes to indicate that we refer to the enclosed passage as a concept, 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 a formal notion in this monograph.

We also use italics to distinguish the formal objects from the words in the English language; furthermore, boldface printing is used to indicate formal operators.

0.3 Notions related to "asynchronous"

In this section we present terms that have been used in literature to refer to asynchronous communication design; we have included a lot of references which can be a starting point for exploring this area. Readers familiar with the research in this area may want to continue reading in section 0.4. At Eindhoven University of Technology a public bibliography on asynchronous communication has been set up. A compressed version of the bibliography file is available for anonymous ftp on Internet from <ftp.win.tue.nl> (address: [131.155.70.100]) as file async.bib.Z in directory /pub/tex. All communication concerning this library can be sent to the corresponding e-mail address:

<async-bib@win.tue.nl>

Many people have been concerned with notions that are related to delay-insensitivity. In the literature one encounters a variety of terms: asynchronous, speed-independent, self-timed, delay-safe, delay-insensitive, delay-independent. Although distinct terms are used, people are dealing with related intuitive notions. Attempts have been made to formalize these notions stressing distinct characteristics. Furthermore, the same term has been used by different people to indicate different aspects of the intuitive notions.

The term asynchronous arose to distinguish between synchronous, e.g. globally clocked, and not synchronous, e.g. locally clocked or not clocked, systems, cf. [Muller – Bartky 59, Unger 69, Rosenberger 69, Keller 75, Molnar – Fang 81, Dill – Clarke 85, Molnar 86, Brzozowski – Seger 89, Brzozowski – Ebergen 89, Yoeli 87]. In [Josephs – Hoare – Jifeng 89] Josephs, Hoare, and Jifeng have introduced asynchronous processes in CSP, cf. [Hoare 85]. Muller, cf. [Miller 65], Keller, cf. [Keller 74], Fang and Molnar, cf. [Fang – Molnar 83], and Dill, cf. [Dill 88], use the term "speed-independent", and Seitz is among others concerned with "self-timed" systems, cf. [Seitz 79, Martin 85b, Yakovlev 85, Greenstreet – Williams – Staunstrup 88]. Van de Snepscheut and Martin both use "delay-insensitive". They stress the internal communication, cf. [van de Snepscheut 85, Martin 86]; 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 Macromodules, cf. [Molnar – Fang – Rosenberger 85, Clark – Molnar 74, Molnar – Fang 81, Rosenberger – Molnar – Chaney – Fang 88]. The internal communication is, generally, not delay-insensitive. Based upon the latter approach several formalizations have emerged, cf. [Udding 84, Schols 85, Verhoeff 85, Black 86, Ebergen 87].

Udding was the first to capture delay-insensitivity formally. He has presented a set of rules, i.e. predicates on trace structures, that are necessary and sufficient for delay-insensitivity, cf. [Udding 84]. Udding is concerned with the communication behavior of components rather than with the communication in channels. He distinguishes four classes; the largest class he has called the "delay-insensitive class", see also chapter 5 and subsection 7.0.1.

Within the study of asynchronous communication design the multiple use of terms has led to argument and confusion. In this monograph, see chapter 5, we will work within the area "delay-insensitive communication", see [Udding 84].

0.4 Delay-insensitivity

Restricting communication to delay-insensitive communication turns out to reduce the class of implementable specifications of circuits. Many questions arise, e.g.:

- what are the limitations of delay-insensitive communication?
- can delay-insensitive communication be integrated with more synchronized forms of communication?
- is any liveness property implementable when using delay-insensitive communication?

In this monograph we address the first two questions extensively. Regarding the third question, it has been argued that liveness properties are not expressable using finite trace theory. We have shown that it is possible to express some liveness properties in finite trace theory, e.g. absence of ambiguous quiescence hazard, cf. "absence of unspecified termination hazard" in [Schols88]; in this monograph ambiguous quiescence hazard is presented as an example of the transformation technique shown in chapter 3.

Seitz argues that a strict protocol of signaling conventions has to be imposed throughout a system in order to deal with the complexity of the design, cf. [Seitz80]. We agree with him. On the other hand, confining oneself to such a restriction may make the design problem fundamentally unsolvable or require unacceptable penalties in cost, performance, manufacturability, or testability. We would like to know whether our inability to find an acceptable solution for such a problem, is fundamentally due to the problem itself or to a possibly too severe restriction that we imposed and that perhaps should be relaxed. In chapters 4 and 5 we present tools that help to answer this question.

0.5 Proofs

Within this monograph we present formal statements in theorems, lemmas, and properties. We present properties without formal proofs, since the proofs of them are either trivial, easy, presented elsewhere, or analogous to other proofs; we do give hints when this is appropriate. The proofs of lemmas and theorems are presented in appendix A; this is done in order not to interrupt the flow of the discourse by the rather technical proofs. Theorems represent the formal conclusions drawn in this monograph; lemmas are intended for local use within the context of this monograph only.

1

Formalism and notation

In this chapter we present some tools and notational conventions that we use in the remainder of this monograph.

1.0 Sets

In this monograph a *set* is denoted by a pair of curly brackets. The elements of a set are listed between these brackets. The elements are separated from each other by commas. We also use quantification to denote sets, see section 1.2.

We denote the *empty set* by "$\varnothing$". Between an element and a set there exists a binary relation, viz. "*is an element of*"; this relation is denoted by the infix operator "$\in$". The negation of this relation, i.e. the binary relation "*is not an element of*", is denoted by the infix operator "$\notin$".

example 1.0

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

$0 \in \{0,1,2\}$

$4 \notin \{0,1,2\}$

end of example

The *intersection* of two sets is denoted by the infix operator "$\cap$". The *union* of two sets is denoted by the infix operator "$\cup$". The binary relation "*subset*" is denoted by the infix operator "$\subseteq$".

example 1.1

$\{2,3\} \cap \{2,4\} = \{2\}$

$\{2,3\} \cup \{2,4\} = \{2,3,4\}$

$\{2,4\} \subseteq \{2,3,4\}$

$\{2,3,4\} \subseteq \{2,3,4\}$

end of example

The binary relation "*proper subset*" is denoted by the infix operator "$\subset$". For sets M and N, $M \subset N$ is equal to $(M \subseteq N) \wedge (M \neq N)$. For sets M and N, we denote the *asymmetric set difference* of M and N by $M \backslash N$. In definition 1.2 we use quantification to denote the set that is defined as $M \backslash N$; we explain this notation in section 1.2.

definition 1.2 *asymmetric set difference*

For sets M and N,

$$M \backslash N \stackrel{\text{def}}{=} \{m : m \in M \wedge m \notin N : m\}.$$

end of definition

For sets M and N, the *symmetric set difference* of M and N is denoted by $M \div N$.

definition 1.3 *symmetric set difference*

For sets M and N,

$$M \div N \stackrel{\text{def}}{=} (M \backslash N) \cup (N \backslash M).$$

end of definition

example 1.4

$\{2,3\} \backslash \{2,4\} = \{3\}$

$\{2,3\} \div \{2,4\} = \{3,4\}$

end of example

The set of *natural numbers* is denoted by $\mathbb{N}$; in this monograph, zero is a natural number. The set of all positive natural numbers is denoted by $\mathbb{N}^+$.

property 1.5

(i) $0 \in \mathbb{N}$

(ii) $\mathbb{N}^+ = \mathbb{N} \backslash \{0\}$

end of property

1.1 Operators

We assume that the reader is familiar with the following operators in propositional calculus: *equality*, denoted by "$=$", *inequality*, denoted by "$\neq$", *negation*, denoted by "$\neg$", *conjunction*, denoted by "$\wedge$", *disjunction*, denoted by "$\vee$", and *implication*, denoted by "$\Rightarrow$". The disjunction is inclusive, i.e. $x \vee y$ does not imply $x \neq y$.

1.1.0 Priority of operators

We define the *priority* of operators in order to save on parentheses. To do this we have grouped the operators; within each group all operators have equal *binding power*. The groups of operators are listed in table 1.0 in order of increasing binding power.

$=$	$\neq$			
$\wedge$	$\vee$	$\Rightarrow$		
$\in$	$\notin$	$\subseteq$	$\subset$	
$\setminus$	$\div$	$\cap$	$\cup$	$\restriction$
all other operators that have at least two parameters				
all unary operators				
catenation				

table 1.0
Priority of operators in order of increasing binding power.

As a consequence, catenation has the highest binding power; equality and inequality have the lowest binding power.

1.2 Quantification

In order to denote quantification, we need a variable binding construct. For such a construct we use a slightly unconventional notation. For instance, *universal quantification*, i.e. generalized conjunction, is denoted by

$$(\mathbf{A}\, l : R : E),$$

where **A** is the quantifier, l is the list of bound variables, R is the predicate that delineates the range of the variables, and E is the quantified expression. Both R and E will, in general, contain variables from l. Analogously, we denote *existential quantification*, i.e. generalized disjunction, by

$$(\mathbf{E}\, l : R : E).$$

Furthermore, we may use *quantification* to denote sets:

$$\{l : R : e\},$$

where e denotes an element of the set.

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

example 1.6

$(\mathbf{A}\, i : 6 \leq i < 9 : P_i)$ is equal to $P_6 \wedge P_7 \wedge P_8$.

$(\mathbf{E}\, i, j : (2 \leq i \leq 5) \wedge EVEN(j) \wedge (i = j) : P_i)$ is equal to $P_2 \vee P_4$.

$\{i : 2 \leq i \leq 4 : i^2\}$ is equal to $\{4, 9, 16\}$.

end of example

1.3 Denotation of proofs

Proofs are often split into a number of steps. For instance, for expressions E, F, and G, we can prove $E \Rightarrow G$ by arguing that $E = F$ and $F \Rightarrow G$. The sameness of the two occurrences of F is essential for the argument that the total proof is correct. To establish this sameness, a string comparison is needed. In order to prevent that the reader has to perform such comparisons, we denote proofs like this in the following way:

$$
\begin{array}{ll}
 & E \\
= & \quad \{ \text{ hint why } E = F \ \} \\
 & F \\
\Rightarrow & \quad \{ \text{ hint why } F \Rightarrow G \ \} \\
 & G
\end{array}
$$

In an analogous way, we denote the proof of $A \subseteq C$ that consists of the steps $A = B$ and $B \subseteq C$. This denotation of proofs is called *hint calculus*. It has been adopted from [Dijkstra – Feijen 88].

1.4 Trace theory

When we refer to *trace theory* in this monograph, we mean the trace theory that has been developed at Eindhoven University of Technology by Rem and others, cf. [Rem – van de Snepscheut – Udding 83, van de Snepscheut 85, Kaldewaij 86, Rem 85]. Trace theory is a tool that has been developed to formalize communication. In this section we present the trace theory notions that are used in this monograph. In subsection 1.4.3 we present the notions that we have added to the notions that exist in trace theory. For a detailed overview of trace theory we refer to [Kaldewaij 86]. In subsection 1.4.4 we present a notational convention that may make it easier to appreciate trace theory.

remark 1.7

Mazurkiewicz, cf. [Mazurkiewicz 85], has developed a formalism that is also called trace theory. Mazurkiewicz's trace theory differs from our trace theory. Mazurkiewicz's traces correspond to equivalence classes over our traces.

end of remark

In our trace theory all traces have finite length. For this reason it is also called finite trace theory. Finite trace theory has been extended by Van Horn, cf. [Van Horn 86], and Black, cf. [Black 86], with infinite traces; this extension is used by them in order to deal with liveness properties. Although liveness properties are not a primary concern in this monograph, we use a liveness

property as an example of a correctness concern in section 3.4. From this we conclude that some liveness properties can be expressed in finite trace theory.

1.4.0 Basic notions of trace theory

We assume the existence of a finite set Ω; Ω is called the *universe*. The elements of Ω are called *symbols*. We assume that Ω is large enough, i.e. we will not run out of symbols. A subset of Ω is called an *alphabet*. A sequence of symbols is called a *trace*. A set of traces is called a *trace set*. The sequence containing no symbols is denoted by ε; trace ε is called the *empty trace*. We link sequences by *catenating* them. Catenation is denoted by juxtaposition. In trace theory the noun "concatenation" is sometimes used instead of "catenation", cf. [Kaldewaij86].

The set of all finite-length sequences of symbols chosen from an alphabet is called the *Kleene-closure* of this alphabet.

definition 1.8 *Kleene-closure of alphabet*

For alphabet A, the trace set that is the Kleene-closure of A is denoted by A^*; it is defined recursively by:

(i) $\varepsilon \in A^*$

(ii) $(\mathbf{A}\, s, a : s \in A^* \wedge a \in A : sa \in A^*)$

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

end of definition

Notice that $\varnothing^* = \{\varepsilon\}$. Furthermore, traces are elements of Ω^*. We extend definition 1.8, "Kleene-closure of alphabet", to trace sets.

definition 1.9 *Kleene-closure of trace set*

For trace set S, the trace set that is the Kleene-closure of S is denoted by S^*; it is defined recursively by:

(i) $\varepsilon \in S^*$

(ii) $(\mathbf{A}\, s, t : s \in S^* \wedge t \in S : st \in S^*)$

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

end of definition

We define the binary operation *prefix* on traces.

definition 1.10 *prefix*

For traces s and t, s is called a prefix of t, denoted by s **prefix** t, if and only if

$$(\mathbf{E}\, u : u \in \Omega^* : su = t)$$

end of definition

In trace theory the symbol "$\leq$" has been used to denote the operation "prefix", cf. [Kaldewaij86]; since the operator "$\leq$" has been used in literature to denote many different operations, we prefer to use **prefix** to denote the operation prefix.

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

definition 1.11 *prefix-closure of trace set*

For trace set S, **pref** S denotes the trace set that contains all prefixes of S:

$$\mathbf{pref}\, S \stackrel{\text{def}}{=} \{s, t : (s \,\mathbf{prefix}\, t) \wedge (t \in S) : s\}$$

end of definition

We call a trace set S *prefix-closed* if and only if $S = \mathbf{pref}\, S$.

We denote the *length* of trace t by $\mathit{l}t$.

definition 1.12 *length of trace*

We define the length of a trace recursively by:

(i) $l\varepsilon \stackrel{\text{def}}{=} 0$

(ii) for trace t and symbol a,

$$lta \stackrel{\text{def}}{=} lt + 1$$

end of definition

For trace t and alphabet A we denote the *projection* of t on A by $t{\restriction}A$.

definition 1.13 *projection of trace*

We define projection of a trace on an alphabet A recursively by:

(i) $\varepsilon{\restriction}A \stackrel{\text{def}}{=} \varepsilon$

(ii) for trace t and symbol a such that $a \in A$,

$$ta{\restriction}A \stackrel{\text{def}}{=} (t{\restriction}A)a$$

(iii) for trace t and symbol a such that $a \notin A$,

$$ta{\restriction}A \stackrel{\text{def}}{=} t{\restriction}A$$

end of definition

We extend the definition of projection to trace sets.

definition 1.14 *projection of trace set*

For trace set S and alphabet A,

$$S\restriction A \stackrel{\text{def}}{=} \{t : t\in S : t\restriction A\}$$

end of definition

In traces, occurrences of symbols are counted from the left to the right. As a consequence, the first occurrence of a symbol in a trace is the left most occurrence of this symbol in this trace. For trace t and symbol a we denote the *number of occurrences* of a in t by $\#_a t$.

definition 1.15 *number of occurrences*

We define the number of occurrences of a symbol a in a trace t by:

$$\#_a t \stackrel{\text{def}}{=} l(t\restriction\{a\}).$$

end of definition

We define the notion *bag*.

definition 1.16 *bag*

A bag, say B, is a set of pairs such that

$$B=\{a : a\in\Omega : (a, f(a))\} \qquad \text{for some function } f : \Omega\rightarrow\mathbb{N}.$$

end of definition

In definition 1.16, "bag", for every symbol a, $f(a)$ is the number of occurrences of a in the bag B. In order to avoid a cumbersome notation, we abbreviate the denotation of a bag, say B, to $\{a, n : (a, n)\in B \wedge n>0 : (a, n)\}$.

We define the *bag of a trace*:

definition 1.17 *bag of trace*

For trace t, **bag** t denotes the bag of t:

$$\textbf{bag}\, t \stackrel{\text{def}}{=} \{a : a\in\Omega : (a, \#_a t)\}$$

end of definition

1.4.1 Trace structures

A *trace structure* is an ordered pair $< A, S >$, in which A denotes an alphabet and S denotes a trace set satisfying $S \subseteq A^*$. For trace structure T, $\mathbf{a}T$ denotes the *alphabet of trace structure* T, and $\mathbf{t}T$ denotes the *trace set of trace structure* T.

We define the partial order *inclusion* on trace structures.

definition 1.18 *trace structure inclusion*

For trace structures T and U, we say that T is included in U, denoted by $T \subseteq U$, if and only if

$$(\mathbf{a}T = \mathbf{a}U) \wedge (\mathbf{t}T \subseteq \mathbf{t}U)$$

end of definition

Of course, the *proper inclusion* $T \subset U$ equals $(\mathbf{a}T = \mathbf{a}U) \wedge (\mathbf{t}T \subseteq \mathbf{t}U) \wedge (\mathbf{t}T \neq \mathbf{t}U)$. We extend the definition of prefix-closure to trace structures.

definition 1.19 *prefix-closure of trace structure*

For trace structure T, $\mathbf{pref}T$ denotes the trace structure that is the prefix-closure of T:

$$\mathbf{pref}T \stackrel{\text{def}}{=} < \mathbf{a}T, \mathbf{pref}(\mathbf{t}T) >$$

end of definition

We call a trace structure T *prefix-closed* if and only if $T = \mathbf{pref}T$. We call a trace structure *nonempty* if and only if its trace set is nonempty.

property 1.20

For prefix-closed trace structure T,

$$(\varepsilon \in \mathbf{t}T) = (T \text{ is nonempty})$$

end of property

We often refer to prefix-closed trace structures that contain ε in their trace set. Using property 1.20 we call such a trace structure a nonempty and prefix-closed trace structure.

For trace structures with equal alphabets we define their *intersection*.

definition 1.21 *intersection of trace structures*

For trace structures T and U such that $\mathbf{a}T = \mathbf{a}U$, the intersection of T and U, denoted by $T \cap U$, is defined by

$$T \cap U \stackrel{\text{def}}{=} < \mathbf{a}T \cap \mathbf{a}U, \mathbf{t}T \cap \mathbf{t}U >$$

end of definition

Analogously, for trace structures with equal alphabets we define their *union*.

definition 1.22 *union of trace structures*

For trace structures T and U such that $\mathbf{a}T = \mathbf{a}U$, the union of T and U, denoted by $T \cup U$, is defined by

$$T \cup U \stackrel{\text{def}}{=} < \mathbf{a}T \cup \mathbf{a}U , \mathbf{t}T \cup \mathbf{t}U >$$

end of definition

We extend the definition of *projection* to trace structures.

definition 1.23 *projection of trace structure*

For trace structure T and alphabet A,

$$T \restriction A \stackrel{\text{def}}{=} < \mathbf{a}T \cap A , \mathbf{t}T \restriction A >$$

end of definition

For trace structures T and U we denote their *weave* by $T \mathbf{w} U$. $T \mathbf{w} U$ is a trace structure. We consider traces t, $t \in \mathbf{t}T$, and u, $u \in \mathbf{t}U$, that are equal w.r.t. the common symbols, i.e. $t \restriction (\mathbf{a}T \cap \mathbf{a}U) = u \restriction (\mathbf{a}T \cap \mathbf{a}U)$. Traces t and u are 'merged' into one or more traces of $\mathbf{t}(T \mathbf{w} U)$; the common symbols are not duplicated by this 'merging'. All pairs of traces t and u that satisfy $t \restriction (\mathbf{a}T \cap \mathbf{a}U) = u \restriction (\mathbf{a}T \cap \mathbf{a}U)$ are 'merged' in this way.

definition 1.24 *weave*

For trace structures T and U,

$$T \mathbf{w} U \stackrel{\text{def}}{=} < \mathbf{a}T \cup \mathbf{a}U , \{ s : s \in (\mathbf{a}T \cup \mathbf{a}U)^* \wedge s \restriction \mathbf{a}T \in \mathbf{t}T \wedge s \restriction \mathbf{a}U \in \mathbf{t}U : s \} >$$

end of definition

In example 1.25 we give examples of the weave of trace structures.

example 1.25

(i) $< \{a , b\} , \{\varepsilon , a , ab , aba\} > \mathbf{w} < \{b , c\} , \{\varepsilon , b , bc\} >$
$= < \{a , b , c\} , \{\varepsilon , a , ab , aba , abc , abac , abca\} >$

(ii) $< \{a , b , d\} , \{\varepsilon , b , d , ba\} > \mathbf{w} < \{b , c , d\} , \{\varepsilon , b , d , dc\} >$
$= < \{a , b , c , d\} , \{\varepsilon , b , d , ba , dc\} >$

end of example

property 1.26 *weaving is symmetric*

For trace structures T and U,

$$T \mathbf{w} U = U \mathbf{w} T$$

end of property

For trace structures T and U such that $\mathbf{a}T \cap \mathbf{a}U = \varnothing$, the trace set of $T \mathbf{w} U$ consists of all traces that are interleavings of a trace of $\mathbf{t}T$ and a trace of $\mathbf{t}U$. Property 1.27 shows that weaving is equal to intersection if the alphabets of the trace structures are equal.

property 1.27

For trace structures T and U such that $\mathbf{a}T = \mathbf{a}U$,

$$T \mathbf{w} U = T \cap U$$

end of property

The weave of nonempty prefix-closed trace structures is a nonempty prefix-closed trace structure:

property 1.28

For nonempty prefix-closed trace structures T and U,

$T \mathbf{w} U$ is nonempty and prefix-closed.

end of property

For trace structures T and U we denote their *blend* by $T \mathbf{b} U$. $T \mathbf{b} U$ is a trace structure, viz. the projection of $T \mathbf{w} U$ on the non-common symbols.

definition 1.29 *blend*

For trace structures T and U,

$$T \mathbf{b} U \stackrel{\text{def}}{=} (T \mathbf{w} U) \upharpoonright (\mathbf{a}T \div \mathbf{a}U)$$

end of definition

In example 1.25 we considered the weave of some trace structures. Examples of the blend of these same trace structures are given in example 1.30.

example 1.30

(i) $< \{a, b\}, \{\varepsilon, a, ab, aba\} > \mathbf{b} < \{b, c\}, \{\varepsilon, b, bc\} >$
$= < \{a, c\}, \{\varepsilon, a, aa, ac, aac, aca\} >$

(ii) $< \{a, b, d\}, \{\varepsilon, b, d, ba\} > \mathbf{b} < \{b, c, d\}, \{\varepsilon, b, d, dc\} >$
$= < \{a, c\}, \{\varepsilon, a, c\} >$

end of example

1.4.2 State graphs

We often denote a nonempty prefix-closed trace set by a *state graph* (i.e. a simple, arc-labeled, directed graph) that is deterministic and minimal, cf. [Kaldewaij 86]. The nodes of the graph are the *states*; the arcs of the graph are the *transitions*. The state, to which trace t corresponds, is denoted by $[t]$. As a consequence, $[\varepsilon]$ denotes the *initial state*. Each path starting in $[\varepsilon]$ yields a trace by catenating the labels of the arcs on that path as they are traversed. If a state graph has a finite number of states, it is called *regular*. In a diagram of a regular state graph the initial state is indicated by a fat dot, see figure 1.1.

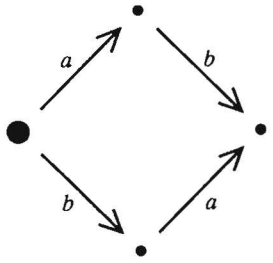

figure 1.1
State graph of trace set $\{\varepsilon, a, b, ab, ba\}$.

A state graph can also denote a nonempty prefix-closed trace structure, say T, if every symbol of $\mathbf{a}T$ occurs in at least one of the traces of $\mathbf{t}T$. In that case the alphabet of the trace structure consists of all symbols that occur as a label of some arc in the state graph. If the state graph in figure 1.1 is used to denote a trace structure, it denotes $< \{a, b\}, \{\varepsilon, a, b, ab, ba\} >$.

Often we present state graphs in which two states are connected by two arcs that point in opposite directions and have the same label, see figure 1.2.

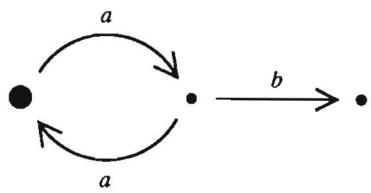

figure 1.2
A state graph.

We abbreviate such a pair of arcs by replacing these two arcs by one bidirectional arc with the same label, see figure 1.3.

figure 1.3
Abbreviated diagram of the state graph of figure 1.2.

As a consequence, the diagrams in figure 1.2 and figure 1.3 are diagrams of the same state graph.

1.4.3 Extensions of trace theory

In this subsection we introduce two extensions of trace theory that are used in this monograph.

1.4.3.0 The bipartitions alphbip and iobip

We introduce the notion *alphbip*. An alphbip is an unordered pair of disjoint sets of symbols. The union of these sets is called the *alphabet of the alphbip*; an alphbip is a bipartition of its alphabet. Given alphbip D, the alphabet of D is denoted by $\mathbf{a}D$. Given two disjoint sets of symbols, say A and B, the alphbip of which A and B are the parts is denoted by $A \oplus B$.

property 1.31 $\oplus$ *is symmetric*

For two disjoint sets, A and B, of symbols,

$$A \oplus B = B \oplus A$$

end of property

definition 1.32 *intersection of alphbip and alphabet*

For two disjoint sets, A and B, of symbols, and alphabet C, the intersection of alphbip $A \oplus B$ with C is defined by:

$$(A \oplus B) \cap C \stackrel{\text{def}}{=} (A \cap C) \oplus (B \cap C)$$

end of definition

For symbol a and alphbip D such that $a \in \mathbf{a}D$, we denote the alphabet of symbols in $\mathbf{a}D$ that are in the *same part of alphbip* D as a by $\mathbf{spa}(a, D)$; we denote the alphabet of symbols in $\mathbf{a}D$ that are in the *other part of alphbip* D than a by $\mathbf{opa}(a, D)$.

property 1.33

For alphbip D,

$$(\mathbf{A}\, a : a \in \mathbf{a}D :\ D = \mathbf{spa}(a, D) \oplus \mathbf{opa}(a, D)).$$

end of property

We also introduce the notion *iobip*. An iobip is a pair of disjoint sets of symbols, viz. the *input alphabet of the iobip* and the *output alphabet of the iobip*. The union of these sets is called the *alphabet of the iobip*; an iobip is an ordered bipartition of its alphabet. Given iobip F, the alphabet of F is denoted by $\mathbf{a}F$. The input alphabet of F is denoted by $\mathbf{i}F$; the output alphabet of F is denoted by $\mathbf{o}F$. For iobip F we define its *reflection*, which is denoted by $\overline{F}$: $\mathbf{i}\overline{F} \stackrel{\text{def}}{=} \mathbf{o}F$ and $\mathbf{o}\overline{F} \stackrel{\text{def}}{=} \mathbf{i}F$. The reflection of an iobip is an iobip.

1.4.3.1 Reduction operator

We introduce the function **redts**; this function reduces the trace set of a trace structure by removing certain traces. The motivation for the introduction of this operator **redts** can only be provided in the context of the following chapters. Until there, the reader may not fully appreciate it.

definition 1.34 **redts**

For trace structure T, alphabet A, and trace set S, we define trace structure $\mathbf{redts}(T, A, S)$ by:

$$\mathbf{redts}(T, A, S) \stackrel{\text{def}}{=}$$
$$< \mathbf{a}T, \mathbf{t}T \setminus \{x, y, w : x \in (\mathbf{a}T)^* \wedge y \in (\mathbf{a}T \setminus A)^* \wedge xy \in (\mathbf{t}T \cap S) \wedge w \in (\mathbf{a}T)^* : xw\} >$$

end of definition

In definition 1.34, "**redts**", not only every trace (xy) in $\mathbf{t}T \cap S$ is removed from $\mathbf{t}T$, but also all prefixes (x) of such a trace (xy) that differ from it (xy) by a sequence (y) of symbols that are not in A; to ensure the prefix-closedness of $\mathbf{t}(\mathbf{redts}(T, A, S))$, all traces (xw) of which a prefix (x) is removed are removed, too. For the necessity of the intersection with $\mathbf{t}T$ in definition 1.34, "**redts**", we refer to example 1.43. The role of alphabet A is illuminated by property 1.35: every trace in $\mathbf{t}T \cap S$ causes the elimination from $\mathbf{t}T$ of a trace that contains a symbol in A.

property 1.35

For prefix-closed trace structure T, alphabet A, and trace set S such that $\varepsilon \in \mathbf{t}(\mathbf{redts}(T, A, S))$,

$$(\mathbf{A}\, s : s \in (\mathbf{t}T \cap S) : (\mathbf{E}\, x, a : x \in (\mathbf{a}T)^* \wedge a \in A \wedge xa \,\mathbf{prefix}\, s : x \in \mathbf{t}(\mathbf{redts}(T, A, S)) \wedge xa \notin \mathbf{t}(\mathbf{redts}(T, A, S)))$$

end of property

In property 1.36 we present a generalization of property 1.35.

property 1.36

For prefix-closed trace structure T, alphabet A, and trace sets S and R such that $\varepsilon \in \mathbf{t}(\mathbf{redts}(T, A, S))$ and $R = (\mathbf{t}T \setminus \mathbf{t}(\mathbf{redts}(T, A, S)))$,

$$(\mathbf{A}\, r : r \in R : (\mathbf{E}\, x, a : x \in (\mathbf{a}T)^* \wedge a \in A \wedge xa \,\mathbf{prefix}\, r : x \in \mathbf{t}(\mathbf{redts}(T, A, S)) \wedge xa \notin \mathbf{t}(\mathbf{redts}(T, A, S)))$$

end of property

remark 1.37

Trace x and symbol a in the existential quantification in properties 1.35 and 1.36 do not only exist: they are also unique.

end of remark

In order to distinguish between statements about formal objects in definitions, properties, lemmas, and theorems and statements about specific instantiations of such objects in examples, we index the instantiations in examples with natural numbers. We refer to the indexed instantiations locally: in the chapter in which they occur.

example 1.38

We consider prefix-closed trace structure T_0, alphabet A_0, and trace sets R_0 and S_0; they are defined by:

$$T_0 \stackrel{\text{def}}{=} <\{a,b\},\{\varepsilon,a,ab\}>,$$
$$A_0 \stackrel{\text{def}}{=} \{a\},$$
$$R_0 \stackrel{\text{def}}{=} \{a\},$$
$$S_0 \stackrel{\text{def}}{=} \{ab\}.$$

From definition 1.34, "**redts**", follows $\mathbf{redts}(T_0,A_0,R_0)=<\{a,b\},\{\varepsilon\}>$. We also see that $\mathbf{redts}(T_0,A_0,S_0)=<\{a,b\},\{\varepsilon\}>$, since trace a is eliminated from $\mathbf{t}T_0$ because $b\in(\mathbf{a}T_0\backslash A_0)^*$ and $ab\in(\mathbf{t}T_0\cap S_0)$.

end of example

The following properties follow from definition 1.34, "**redts**".

property 1.39 **redts** *preserves prefix-closedness*

For prefix-closed trace structure T, alphabet A, and trace set S,

$\mathbf{redts}(T,A,S)$ is prefix-closed.

end of property

property 1.40

For nonempty, prefix-closed trace structure T, alphabet A, and trace set S,

$$(\mathbf{A}\,s : s\in(\mathbf{t}T\cap S) : l(s\restriction A)>0) = (\varepsilon\in\mathbf{t}(\mathbf{redts}(T,A,S)))$$

end of property

property 1.41

For trace structure T, alphabet A, and trace sets R and S,

$$\mathbf{redts}(T,A,R\cup S) = \mathbf{redts}(\mathbf{redts}(T,A,R),A,S)$$

end of property

property 1.42

For trace structure T, alphabet A, and trace set S such that $\mathbf{t}T \cap S = \varnothing$,

$$\mathbf{redts}(T, A, S) = T$$

end of property

In example 1.43 we illustrate the necessity of the intersection with $\mathbf{t}T$ in definition 1.34, "**redts**".

example 1.43

We consider prefix-closed trace structure T_1, alphabet A_1, and trace set S_1; they are defined by:

$$T_1 \stackrel{\text{def}}{=} < \{a, b\}, \{\varepsilon, a\} >,$$

$$A_1 \stackrel{\text{def}}{=} \{a\},$$

$$S_1 \stackrel{\text{def}}{=} \{ab\}.$$

We are interested in $\mathbf{redts}(T_1, A_1, S_1)$. If the intersection with $\mathbf{t}T_1$ in definition 1.34, "**redts**", is not present, then trace a would be removed when reducing the trace set of T_1, since $ab \in S_1$ and $b \in (\mathbf{a}T_1 \backslash A_1)^*$. We see, however, that there is no need to remove trace a from $\mathbf{t}T_1$, since $ab \notin \mathbf{t}T_1$ anyway.

end of example

1.4.4 Notational convention

Lower case letters near the beginning of the Latin alphabet are symbols; when they are used as variables, they denote symbols. Lower case letters near the end of the Latin alphabet denote traces. Capital letters are used to denote alphabets, alphbips, trace sets, and trace structures.

Boldface lower case operators are used in the trace theory formalism; this does allow them to range over objects in the Communication Model. Boldface upper case operators are used in the Communication Model; this does allow them to range over objects in the trace theory formalism.

2

Communication Model

In this chapter we introduce the *Communication Model*. By introducing this model we achieve a separation of concerns between the interpretation of the underlying physics and the use of the trace theory formalism. We do not interpret any notions of trace theory in the underlying physics directly: we interpret them in our Communication Model. The importance of establishing the separation of concerns between the interpretation of the physical model and the formalism has been recognized previously by others. Van de Snepscheut, see [van de Snepscheut 85], and Udding, see [Udding 84], carefully distinguished trace theory from its mechanistic appreciation. We make this separation of concerns even more explicit by the introduction of our Communication Model.

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 private notion of 'physical reality'. In this chapter we will set down the postulates for our Communication Model. These postulates have been chosen so as to be consistent with at least one class of "physical models" that is used for the design of computing machinery. In this monograph we will not present rigorous arguments for this consistency: we rather discuss the 'reasonableness' of the postulates in one interpretational example, which we will address as "the physical model" in the remainder of this monograph.

The Communication Model is introduced formally in section 2.0. We discuss the relation between our Communication Model and the physical model in section 2.1. In section 2.2 we introduce the trace theory formalism. In section 2.3 we present some examples and in section 2.4 we motivate why we have chosen to make our Communication Model an event-based model.

2.0 Definition of Communication Model

In this section we present the definitions and postulates that form the foundation of our Communication Model. The motivation for choosing these definitions and postulates is provided in section 2.1.

2.0.0 Commports

We assume the existence of a finite set Ψ. The elements of Ψ are called *commports*. Ψ is partitioned into two parts: Ψ^o, the set of *output commports*, and Ψ^i, the set of *input commports*. Of course, the set of commports is disjunct with the set Ω of symbols, which has been introduced in subsection 1.4.0.

postulate 2.0

(i) $\Psi = \Psi^o \cup \Psi^i$

(ii) $\Psi^o \cap \Psi^i = \varnothing$

(iii) $\Psi \cap \Omega = \varnothing$

end of postulate

For output commport α and input commport β, we introduce the predicate "α *matches* β", which is denoted by $\alpha MATCH \beta$. We postulate that a commport matches exactly one commport. Matching commports are either "*connected directly*" or "*connected indirectly*".

postulate 2.1

(i) For input commport γ,

$$(\mathbf{E}\,\alpha : \alpha \in \Psi^o : \alpha \mathit{MATCH} \gamma)$$

(ii) for output commports α and β, and input commport γ,

$$(\alpha \mathit{MATCH} \gamma \wedge \beta \mathit{MATCH} \gamma) \Rightarrow (\alpha = \beta)$$

(iii) for output commport α,

$$(\mathbf{E}\,\gamma : \gamma \in \Psi^i : \alpha \mathit{MATCH} \gamma)$$

(iv) for output commport α, and input commports γ and δ,

$$(\alpha \mathit{MATCH} \gamma \wedge \alpha \mathit{MATCH} \delta) \Rightarrow (\gamma = \delta)$$

(v) for commports α and β such that $\alpha \mathit{MATCH} \beta$, either α and β are "connected directly" or α and β are "connected indirectly".

end of postulate

From postulate 2.1 we infer that Ψ^o and Ψ^i have the same number of elements. From the definition of matching commports we infer property 2.2.

property 2.2

For commports α and β,

$$\alpha \mathit{MATCH} \beta \Rightarrow (\alpha \in \Psi^o \wedge \beta \in \Psi^i)$$

end of property

2.0.1 Comminsts and commsigs

The elements of the Cartesian product of Ψ and $\mathbb{N}^+$ are called *comminsts*. The comminst with commport α and positive natural number n is denoted by α_n. If α is an output commport, we call α_n an *output comminst*; if α is an input commport, we call α_n an *input comminst*. A set of comminsts, say Λ, is called an *initial set of comminsts* if, for every comminst in Λ, Λ contains all comminsts with the same commport and a smaller number, see definition 2.3.

definition 2.3 *initial set of comminsts*

A set of comminsts Λ is called an *initial set of comminsts* if and only if

$$(\mathbf{A}\,\alpha, m, n : \alpha_n \in \Lambda \wedge m \in \mathbb{N}^+ \wedge m < n : \alpha_m \in \Lambda)$$

end of definition

The elements of the Cartesian product of Ψ^o, $\mathbb{N}^+$, and Ψ^i, for which the output commport matches the input commport, are called *commsigs*, see definition 2.4:

definition 2.4

For α, β, and n such that $\alpha \in \Psi^o$, $\beta \in \Psi^i$, $n \in \mathbb{N}^+$, and $\alpha \mathit{MATCH} \beta$, the triple (α, n, β) is a *commsig*.

end of definition

Analogously to "initial set of comminsts", we define the predicate "initial" for sets of commsigs. A set of commsigs, say Λ, is called an *initial set of commsigs* if, for every commsig in Λ, Λ contains all commsigs with the same pair of matching commports and a smaller number, see definition 2.5.

definition 2.5 *initial set of commsigs*

A set of commsigs Λ is called an *initial set of commsigs* if and only if

$$(\mathbf{A}\, \alpha, \beta, m, n : (\alpha, n, \beta) \in \Lambda \wedge m \in \mathbb{N}^+ \wedge m < n : (\alpha, m, \beta) \in \Lambda)$$

end of definition

2.0.2 Comminstorders and commsigorders

In order to define comminstorders and commsigorders we need the notion *strict partial order*. A strict partial order is an antireflexive and transitive relation; as a consequence, it is antisymmetric. It is also referred to as an "antireflexive partial order" in literature.

A *comminstorder* is a pair $<\Lambda, \sqsubset>$, in which Λ denotes a finite initial set of comminsts and "$\sqsubset$" is a strict partial order on Λ. For comminstorder ϕ, Λ_ϕ denotes the *set of comminsts of comminstorder* ϕ, and $\sqsubset_\phi$ denotes the *strict partial order of comminstorder* ϕ.

In the strict partial order of a comminstorder, a comminst is preceded by every comminst with the same commport and a smaller number.

postulate 2.6

For comminstorder ϕ,

$$(\mathbf{A}\, \alpha, m, n : \alpha_n \in \Lambda_\phi \wedge m \in \mathbb{N}^+ \wedge m < n : \alpha_m \sqsubset_\phi \alpha_n)$$

end of postulate

For comminstorders we define the *restriction* to an initial set of comminsts:

definition 2.7 *restriction of comminstorder*

For comminstorder ϕ and initial set of comminsts Λ, we denote the comminstorder that is *the restriction of* ϕ *to* Λ by $\phi\restriction\Lambda$; it is defined by:

$$\phi\restriction\Lambda \stackrel{\text{def}}{=} <\Lambda_I, \sqsubset_I>$$

where $\Lambda_I = \Lambda_\phi \cap \Lambda$ and $\sqsubset_I$ is given by:

$$(\mathbf{A}\,\lambda,\mu : : (\lambda\in\Lambda_I \wedge \mu\in\Lambda_I \wedge \lambda\sqsubset_\phi\mu) = (\lambda\sqsubset_I\mu))$$

end of definition

Notice that Λ_I in definition 2.7, "restriction of comminstorder", is an initial set of comminsts.

Analogously to comminstorders, we define commsigorders. A *commsigorder* is a pair $<\Lambda,\sqsubset>$, in which Λ denotes a finite initial set of commsigs and "$\sqsubset$" is a strict partial order on Λ. For commsigorder ϕ, Λ_ϕ denotes the *set of commsigs of commsigorder* ϕ, and $\sqsubset_\phi$ denotes the *strict partial order of commsigorder* ϕ.

In the strict partial order of a commsigorder, a commsig is preceeded by every commsig with the same output commport and a smaller number; notice that commsigs with the same output commport also have the same input commport.

postulate 2.8

For commsigorder ϕ,

$$(\mathbf{A}\,\alpha,\beta,m,n : (\alpha,n,\beta)\in\Lambda_\phi \wedge m\in\mathbb{N}^+ \wedge m<n : (\alpha,m,\beta)\sqsubset_\phi(\alpha,n,\beta))$$

end of postulate

For commsigorders we define the *restriction* to an initial set of commsigs:

definition 2.9 *restriction of commsigorder*

For commsigorder ϕ and initial set of commsigs Λ, we denote the commsigorder that is *the restriction of* ϕ *to* Λ by $\phi\restriction\Lambda$; it is defined by:

$$\phi\restriction\Lambda \stackrel{\text{def}}{=} <\Lambda_I, \sqsubset_I>$$

where $\Lambda_I = \Lambda_\phi \cap \Lambda$ and $\sqsubset_I$ is given by:

$$(\mathbf{A}\,\lambda,\mu : : (\lambda\in\Lambda_I \wedge \mu\in\Lambda_I \wedge \lambda\sqsubset_\phi\mu) = (\lambda\sqsubset_I\mu))$$

end of definition

Notice that Λ_I in definition 2.9, "restriction of commsigorder", is an initial set of commsigs.

2.0.3 Iodirs and modules

An *iodir*, say Φ, is a pair $<\Phi^o, \Phi^i>$, in which Φ^o is a set of output commports and Φ^i is a set of input commports.

postulate 2.10

For iodir Φ,

(i) $\Phi^o \subseteq \Psi^o$

(ii) $\Phi^i \subseteq \Psi^i$

end of postulate

We define the *reflection* of an iodir. The reflection of an iodir is an iodir.

definition 2.11 *reflection of iodir*

For an iodir Φ, the reflection of Φ, which is denoted by $\bar{\Phi}$, is defined by

$$\bar{\Phi}^o \stackrel{\text{def}}{=} \Phi^i$$
$$\bar{\Phi}^i \stackrel{\text{def}}{=} \Phi^o$$

end of definition

A *module*, say Δ, is a pair $<\mathbf{IO}\,\Delta, \mathbf{CB}\,\Delta>$, in which $\mathbf{IO}\,\Delta$ is an iodir and $\mathbf{CB}\,\Delta$ is a set of comminstorders; $\mathbf{IO}\,\Delta$ is called the *iodir of module* Δ, and $\mathbf{CB}\,\Delta$ is called the *communication behavior of module* Δ. Ψ^o_Δ is called the *set of output commports of module* Δ, Ψ^i_Δ is called the *set of input commports of module* Δ. Of course, $\mathbf{IO}\,\Delta = <\Psi^o_\Delta, \Psi^i_\Delta>$. We postulate that no two commports of Δ match, that the *empty comminstorder*, i.e. $<\varnothing,\varnothing>$, is in $\mathbf{CB}\,\Delta$, and that $\mathbf{CB}\,\Delta$ is closed with respect to restriction.

postulate 2.12

For module Δ,

(i) for every output commport $\alpha \in \Psi^o_\Delta$ and input commport $\beta \in \Psi^i_\Delta$,

$$\neg\,(\alpha \mathit{MATCH} \beta)$$

(ii) for every comminstorder $\phi \in \mathbf{CB}\,\Delta$,

$$\Lambda_\phi \subseteq \{\Psi^o_\Delta \cup \Psi^i_\Delta\} \times \mathbb{N}^+$$

(iii) $<\varnothing,\varnothing> \in \mathbf{CB}\,\Delta$,

(iv) for every comminstorder $\phi \in \mathbf{CB}\,\Delta$ and comminst $\lambda \in \Lambda_\phi$ such that $(\mathbf{A}\,\mu : \mu \in \Lambda_\phi : \neg(\lambda \sqsubset_\phi \mu))$,

$$(\phi\restriction(\Lambda_\phi \setminus \{\lambda\})) \in \mathbf{CB}\,\Delta$$

end of postulate

2.0.4 Opdirs and interconnections

An *opdir*, say Ξ, is an unordered pair $<\Xi', \Xi''>$, in which Ξ' and Ξ'' are disjoint sets of input commports. Since an opdir is an unordered pair, the opdirs Ξ, i.e. $<\Xi', \Xi''>$, and $<\Xi'', \Xi'>$ are equal.

postulate 2.13

For opdir Ξ,

(i) $\Xi' \subseteq \Psi^i$

(ii) $\Xi'' \subseteq \Psi^i$

(iii) $\Xi' \cap \Xi'' = \varnothing$

(iv) for opdirs Ξ_1 and Ξ_2 such that $\Xi_1' = \Xi_2''$ and $\Xi_1'' = \Xi_2'$,

$$\Xi_1 = \Xi_2$$

end of postulate

An *interconnection*, say Π, is a pair $<\mathbf{OP}\,\Pi, \mathbf{CM}\,\Pi>$, in which $\mathbf{OP}\,\Pi$ is an opdir and $\mathbf{CM}\,\Pi$ is a set of commsigorders; $\mathbf{OP}\,\Pi$ is called the *opdir of interconnection* Π and $\mathbf{CM}\,\Pi$ is called the *communication of interconnection* Π. We postulate that the *empty commsigorder*, i.e. $<\varnothing,\varnothing>$, is in $\mathbf{CM}\,\Pi$, and that $\mathbf{CM}\,\Pi$ is closed with respect to restriction.

postulate 2.14

For interconnection Π,

(i) $(\mathbf{OP}\,\Pi)' \cap (\mathbf{OP}\,\Pi)'' = \varnothing$

(ii) for every commsigorder $\phi \in \mathbf{CM}\,\Pi$ and commsig $(\alpha, m, \beta) \in \Lambda_\phi$,

$$\beta \in ((\mathbf{OP}\,\Pi)' \cup (\mathbf{OP}\,\Pi)'') \wedge m \in \mathbb{N}^+ \wedge \alpha \mathit{MATCH} \beta$$

(iii) $<\varnothing,\varnothing> \in \mathbf{CM}\,\Pi$,

(iv) for every commsigorder $\phi \in \mathbf{CM}\,\Pi$ and commsig $\lambda \in \Lambda_\phi$ such that $(\mathbf{A}\,\mu : \mu \in \Lambda_\phi : \neg(\lambda \sqsubset_\phi \mu))$,

$$(\phi \restriction (\Lambda_\phi \setminus \{\lambda\})) \in \mathbf{CM}\,\Pi$$

end of postulate

The asymmetry in postulate 2.14(ii) is caused by β being the input commport in (α, m, β) and both $(\mathbf{OP}\,\Pi)'$ and $(\mathbf{OP}\,\Pi)''$ being sets of input commports.

We say that *two commports have the same type with respect to interconnection* Π if either both are in $(\mathbf{OP}\,\Pi)'$ or both are in $(\mathbf{OP}\,\Pi)''$.

2.1 Interpretation of Communication Model

We interpret our Communication Model in the physical model. In the physical model we refer to "mechanisms", "terminals", "wires", and "signals". Mechanisms convey information to each other by exchanging signals: a mechanism sends a signal at one of its terminals; this signal is received by a mechanism at a terminal. Either these two terminals are connected by a wire or they coincide.

In our Communication Model we abstract from voltage levels, transmission times, and the difference between high-going and low-going transitions. Furthermore, we model the sending and reception of signals as point actions, i.e. they have no duration.

2.1.0 Commports

A terminal in the physical model is modeled in our Communication Model by zero or more *commports*. A terminal that can only be used by one mechanism to send signals to one terminal of one other mechanism is modeled by one output commport. Analogously, a terminal that can only be used by one mechanism to receive signals from one terminal of one other mechanism is modeled by one input commport. In general, a terminal, that can be used by a mechanism to send signals to m terminals and to receive signals from n terminals, is modeled by m output commports and n input commports. As a consequence, every commport is either an output commport or an input commport, see postulate 2.0(i) and (ii).

Let a mechanism be able to send a signal from a terminal, say terminal *I*, to one specific terminal, say terminal *II*, of another mechanism. The output commport that models the sending of such a signal at terminal *I* is said to *match* the input commport that models the reception of this signal at terminal *II*. From the way the terminals have been 'split' into commports we infer that every commport matches exactly one other commport, see postulate 2.1. Matching commports that model terminals that coincide are said to be *directly connected*. Matching commports that model terminals that are connected by a wire are said to be *indirectly connected*, see postulate 2.1(v).

remark 2.15

Not allowing one-to-many communication from a commport does not exclude broadcasting or buswire communication from the descriptive power of our Communication Model. We deal with these communication forms by introducing modules for them.

end of remark

2.1.1 Comminsts

The act of sending a signal by a mechanism is modeled by a *comminst*. Let commport α model (the part of) the terminal, say terminal *I*, that is used by a mechanism to send signals to one specific terminal, say terminal *II*, of another mechanism. The act of sending the first signal from terminal *I* to terminal *II* is modeled by α_1, the act of sending the second one by α_2, and so on. In a similar way we denote the act of receiving a signal. We treat comminsts as point actions, i.e. they have no duration.

No second signal can be sent from one terminal to another before the first one has been sent. The same holds for the reception of signals. For this reason we are often interested in sets of comminsts that are closed with respect to the lower numbered comminsts. Such a set was called an *initial set of comminsts* (see definition 2.3).

2.1.2 Comminstorders

An order in which signals are sent and received is modeled by a *comminstorder*. When the sending or reception of a signal causally precedes the sending or reception of another signal, we model this by: a comminst occurs *before* another comminst in a comminstorder. Signals that are sent or received in *parallel* or *concurrently* are modeled in our Communication Model as comminsts that occur *independently* –i.e. there exists no causal relation between the sending or reception (of signals) that they model– in a comminstorder. Notice that in our Communication Model the negation of "before" is "*after* or independently". For this reason, a comminstorder is a strict partial order on a set of comminsts, cf. subsection 2.0.2. Comminsts occur either one before the other or independently; no two comminsts occur together. As a consequence, our Communication Model has an interleaving semantics.

Comminst α_2 models the act of sending (or receiving) the second signal at the part of the terminal that is modeled by commport α, see subsection 2.1.1. If α_2 occurs before β_3 in comminstorder ϕ, then, of course, α_1 occurs before β_3 in comminstorder ϕ. In order not to need to specify this explicitly, we require that $\alpha_1 \sqsubset_\phi \alpha_2$, cf. postulate 2.6. Using postulate 2.6 and the transitivity of strict partial orders, we infer from $\alpha_2 \sqsubset_\phi \beta_3$ that $\alpha_1 \sqsubset_\phi \beta_3$. This motivates why Λ_ϕ is an *initial* set of comminsts, cf. subsection 2.0.2. We have chosen to introduce postulate 2.6 in order to be able to state that $\alpha_i \sqsubset_\phi \beta_j$ in stead of $(\mathbf{A}\, k : 1 \le k \le i : \alpha_k \sqsubset_\phi \beta_j)$.

In our Communication Model we deal with finite behaviors. For this reason, we only consider comminstorders with finite sets of comminsts, cf. subsection 2.0.2.

2.1.3 Modules

A mechanism is modeled in our Communication Model by a *module*. The terminals of this mechanism, that can be used by this mechanism to send signals to another mechanism, are modeled by one or more output commports, cf. subsection 2.1.0. These output commports are the *output commports of the module* that models the mechanism, cf. subsection 2.0.3. Analogously, we define the *input commports of the module* that models the mechanism, cf. subsection 2.0.3. We distinguish output and input commports, while we assume that mechanisms actively send signals but passively undergo the reception of signals: a mechanism controls the production of the signals that it sends, but it has no control over the production of the signals that it receives. This distinction is elaborated on in chapter 3. A comminst that models the sending of a signal by a mechanism is called an *output comminst* of the module that models this mechanism; a comminst that models the reception of a signal by a mechanism is called an *input comminst* of the module that models this mechanism.

We assume that no mechanism sends a signal to itself, cf. postulate 2.12(i). An order in which a mechanism may send or receive signals is modeled by a *comminstorder of the module*. From this follows postulate 2.12(ii). Initially, no signals have been sent or received. This is modeled by the empty comminstorder being a member of the set of comminstorders of the module, cf. postulate 2.12(iii). A comminstorder models a possible behavior of a mechanism. If we omit from such a behavior a signal that has no successors, we are left with another possible behavior of the mechanism. The latter behavior is also modeled by a comminstorder of the module that models this mechanism, cf. postulate 2.12(iv).

We consider a comminst that models the reception of a signal by a mechanism. The mechanism is modeled by a module. We say that the mechanism *accepts* this signal, if this comminst is in accordance with a comminstorder of the module: in this case there is no instance of computation interference. For a formal definition of computation interference we refer to chapter 3.

remark 2.16

We have postulated that the communication behavior of a module is such that the mechanism accepts a signal when the comminst that models this signal is in accordance with a comminstorder of the module. Furthermore, the mechanism may send a signal when the comminst that models this signal is in accordance with a comminstorder of the module. There is no obligation for a mechanism to send a signal, even if it is consistent with a comminstorder of the module.

end of remark

We present some modules in the following examples.

example 2.17

We consider a mechanism that can receive one out of two input signals after which it sends an output signal. The module that models this mechanism has one output commport, say γ, and two input commports, say α and β. This module has five comminstorders, viz.

$<\varnothing, \varnothing>$

$<\{\alpha_I\}, \varnothing>$

$<\{\beta_I\}, \varnothing>$

$<\{\alpha_I, \gamma_I\}, \{\alpha_I \sqsubset \gamma_I\}>$

$<\{\beta_I, \gamma_I\}, \{\beta_I \sqsubset \gamma_I\}>$

end of example

example 2.18

We consider a mechanism that can receive two input signals independently of each other after which it sends an output signal. The module that models this mechanism has one output commport, say γ, and two input commports, say α and β. This module has five comminstorders, viz.

$<\varnothing, \varnothing>$

$<\{\alpha_I\}, \varnothing>$

$<\{\beta_I\}, \varnothing>$

$<\{\alpha_I, \beta_I\}, \varnothing>$

$<\{\alpha_I, \beta_I, \gamma_I\}, \{\alpha_I \sqsubset \gamma_I, \beta_I \sqsubset \gamma_I\}>$

end of example

remark 2.19

It is possible to infer the comminstorders of a module from the causal orderings of signals exchanged by a mechanism: i.e. no temporal ordering of signals has to be taken into account, cf. example 2.20.

end of remark

example 2.20 *"Pure Delay" element*

In the mechanism "Pure Delay" element every output is causally preceded by an input. Let α denote the input commport and let β denote the output commport. For every pair (m,n) such that $0 \le n \le m$, we infer one comminstorder, say ϕ, of the module that models this mechanism: Λ_ϕ is $\{i : 0<i\le m : \alpha_i\} \cup \{j : 0<j\le n : \beta_j\}$ and the transitive closure of $\{k : 0<k\le n : \alpha_k \sqsubset_\phi \beta_k\} \cup \{k : 0<k<m : \alpha_k \sqsubset_\phi \alpha_{k+1}\} \cup \{k : 0<k<n : \beta_k \sqsubset_\phi \beta_{k+1}\}$ defines $\sqsubset_\phi$.

end of example

2.1.3.0 Connected modules

We postulate that modules can be connected in different ways: a direct connection and an indirect connection; furthermore, we consider the general case in which both connection ways are combined: a mixed connection.

Two modules have a *direct connection* if all their matching commports are directly connected, see figure 2.34.

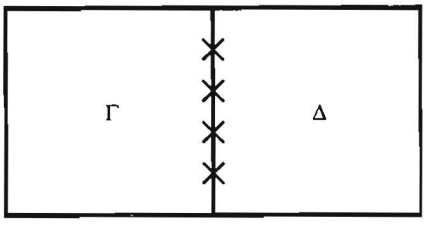

figure 2.0
Direct connection of modules Γ and Δ.

In figures of modules (and components) we indicate the commports of a module (component) by crosses at the boundary of this module (component).

Two modules have an *indirect connection* if all their matching commports are indirectly connected, see figure 2.1.

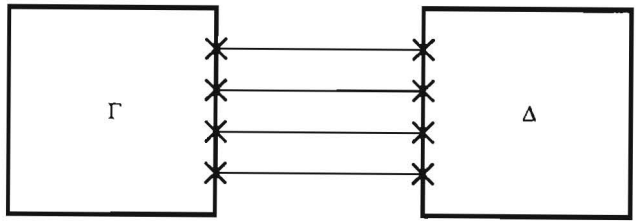

figure 2.1
Indirect connection of modules Γ and Δ.

In the general case, in which some of the matching commports of two modules are directly connected and the others are indirectly connected, we say that the modules have a *mixed connection*, see figure 2.2.

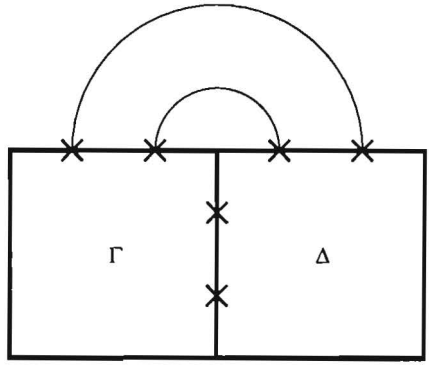

figure 2.2
Mixed connection of modules Γ and Δ.

We discuss directly connected modules in chapter 3; in chapters 4 and 5 we are concerned with indirectly connected modules; we discuss modules that have a mixed connection in chapter 6.

We say that modules have a *closed* connection if every output commport of every module in this connection matches an input commport of another module in this connection, i.e. there is no communication between the connected modules and some environment. Modules that have a connection that is not closed are said to have an *open* connection. We address modules that have a closed connection in chapters 3, 4, and 5. In chapter 6 we deal with modules that either have a closed or an open connection.

2.1.4 Commsigs

A signal travels from the terminal at which it has been sent to the terminal at which it is received. This is modeled by a *commsig*. We do not distinguish between two signals that have been sent from one terminal to one other terminal, while they travel between these terminals. As a consequence, "overtaking of such signals" is a meaningless notion in our model. Furthermore, we assume that every signal that is sent also is received. As a consequence, the first signal that is sent from a terminal, say terminal *I*, to another terminal, say terminal *II*, is the first signal that is received by terminal *II* from terminal *I*. This is modeled in definition 2.4: two commports are used to define a commsig, yet only one number is used. Since signals sent from one specific terminal to one other terminal do not overtake one another, we are often interested in *initial sets of commsigs* (see definition 2.5).

We consider a module, which models a mechanism. A commsig that models a signal that is sent by the mechanism is said to be *sent by the module*. A commsig that models a signal that is received by the mechanism is said to be *received by the module*; a commsig that models a signal that is accepted by the mechanism is said to be *accepted by the module*.

2.1.5 Commsigorders

An order, in which signals that travel between two mechanisms occur, is modeled by a *commsigorder*. When a mechanism has to receive a signal *I* before it sends signal *II*, we say that signal *I* precedes signal *II*. This causality relation is modeled by: a commsig occurs *before* another commsig in a commsigorder. Again, cf. subsection 2.1.2, the negation of "before" is "*after* or *independently*". For this reason, a commsigorder is a strict partial order on a set of commsigs, cf. subsection 2.0.2. And again, commsigs occur either one before the other or independently; we do not model that commsigs occur together.

We argued in subsection 2.1.4 that overtaking of signals that have been sent from one terminal to one other terminal is a meaningless notion in our Communication Model. This is why we are allowed to introduce postulate 2.8. The reason for introducing it is the same one as the reason for introducing postulate 2.6, cf. subsection 2.1.2. Analogously to subsection 2.1.2, we use postulate 2.8 and the transitivity of strict partial orders to motivate that, for commsigorder ϕ, Λ_ϕ is an *initial* set of commsigs, cf. subsection 2.0.2. Furthermore, since we deal with finite behaviors in our Communication Model, we only consider commsigorders with a finite set of commsigs, cf. subsection 2.0.2.

2.1.6 Interconnections

We next consider two mechanisms such that the modules that model these mechanisms have a closed connection. The communication between such two mechanisms is modeled in our Communication Model by an *interconnection*. In such a closed connection every output commport of one module matches one input commport of the other module. For this reason, the input commports suffice to identify all commports of the two modules, the communication between which is modeled by the interconnection, cf. subsection 2.0.4. The amount of delay between the sending of a signal by a mechanism and the reception of this signal by another mechanism is nonnegative. Due to this asymmetric delay we distinguish two directions in an interconnection, which are opposite to each other.

remark 2.21

Unlike the types of comminsts with respect to modules, which have been classified as either input or output, there is no point in classifying commsigs of an interconnection based upon their direction: they merely are distinct.

end of remark

The sets of input commports of the two modules are disjunct, cf. postulate 2.14(i). An order in which signals that are exchanged by two mechanisms happen, is modeled by a *commsigorder* of the interconnection. This is our motivation for postulate 2.14(ii). Initially, no signals have been exchanged. This is modeled by the empty commsig order being a member of the set of commsigorders of the interconnection, cf. postulate 2.14(iii). A commsigorder models a possible exchange of signals between two mechanisms. If we omit from such an exchange a signal that has no successors, we are left with another possible exchange of signals between the two mechanisms. The latter exchange is also modeled by a commsigorder of the interconnection that models the exchange of signals between these two mechanisms, cf. postulate 2.14(iv).

remark 2.22

We do not discuss *observation* nor problems related to observing communication. Although it is possible to discuss some observation issues within our Communication Model, we will not do so in this monograph. As a consequence, our results do not depend upon notions of observation.

end of remark

2.1.6.0 Interconnection between two modules

In this subsection we present a method to *construct the interconnection between two modules that have a closed connection*. This construction method depends on the way in which the commports of these modules are connected. We consider the interconnection, say Π, between two modules, say Γ and Δ. In general, Γ and Δ have a mixed connection.

First, we define an operator that reduces the amount of ordering in a comminstorder. We only keep orderings of the forms $\alpha_i \sqsubset \beta_j$ and $\gamma_m \sqsubset \gamma_n$, in which commports α, β, and γ are such that β is an output commport and that α is not an indirectly connected output commport. In the remainder of this subsection we denote by Ψ_{ic} the set of commports that are *connected indirectly* to their matching commport.

definition 2.23 **REDOC**

For comminstorder ϕ, we define comminstorder **REDOC** ϕ. The set $\Lambda_{\mathbf{REDOC}\,\phi}$ of comminsts is equal to the set Λ_ϕ of comminsts. The set of orderings of **REDOC** ϕ is:

$$\{\alpha, \beta, i, j : \alpha_i \sqsubset_\phi \beta_j \wedge \beta \in \Psi^o \wedge \alpha \notin (\Psi^o \cap \Psi_{ic}) : \alpha_i \sqsubset_{\mathbf{REDOC}\,\phi} \beta_j\}$$
$$\cup \{\gamma, m, n : \gamma_m \sqsubset_\phi \gamma_n : \gamma_m \sqsubset_{\mathbf{REDOC}\,\phi} \gamma_n\}$$

end of definition

Notice that in the definition above β is an output commport and that α is either an input commport or a directly connected output commport. Furthermore, from $\gamma_m \sqsubset_\phi \gamma_n$ follows that $m < n$.

We use the reduction operator **REDOC** to construct the commsigorders of Π out of the comminstorders of Γ and Δ.

Let ϕ_Γ be a comminstorder of Γ, and let ϕ_Δ be a comminstorder of Δ such that for every output commport α and input commport β such that $\alpha \mathit{MATCH} \beta$, if β_j is in the set of comminsts of one of these comminstorders then α_j is in the set of comminsts of the other comminstorder. For every such pair we construct a commsigorder ξ_Π in the following way:

- We define ϕ to be the comminstorder on the union of the sets of comminsts of ϕ_Γ and ϕ_Δ such that the set of orderings of ϕ is the union of the sets of orderings of **REDOC** ϕ_Γ and **REDOC** ϕ_Δ.

- Now, we transform comminstorder ϕ into commsigorder ξ_Π by renaming the comminsts in ϕ. Let α be an output commport of Γ; let β be the input commport of Δ such that α ***MATCH*** β. We rename output comminst α_i into commsig (α, i, β). Analogously, let α be an input commport of Γ; let β be the output commport of Δ such that β ***MATCH*** α. We rename input comminst α_i into commsig (β, i, α).
  In this way, we rename every comminst of ϕ_Γ into a commsig of ξ_Π; analogously, we rename every comminst of ϕ_Δ into a commsig of ξ_Π.
- Commsigorder ξ_Π is the result of this renaming in comminstorder ϕ.

We now define interconnection Π. Its opdir **OP** Π is equal to $<\Psi_\Gamma^i, \Psi_\Delta^i>$; its communication **CM** Π is equal to the union of the set of all commsigorders ξ_Π that can be constructed in the way described above and the set of all commsigorders that are restrictions of such a ξ_Π to an initial set of commsigs. This construction method is demonstrated in example 2.24.

example 2.24

We consider modules Δ_0 and Δ_1. Δ_0 has output commport α and input commports β and γ. Δ_1 has output commports ζ and η and input commport δ. These commports match in the following way: α ***MATCH*** δ, ζ ***MATCH*** β, and η ***MATCH*** γ. α and δ are indirectly connected, ζ and β are indirectly connected, but η and γ are directly connected. As a consequence, Δ_0 and Δ_1 have a closed mixed connection, see figure 2.3.

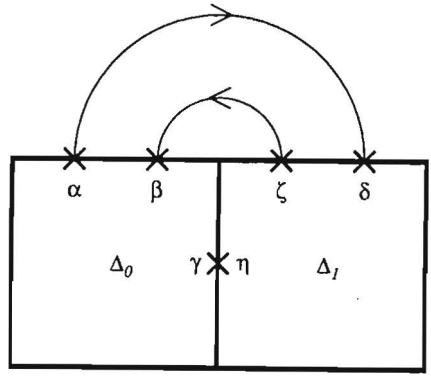

figure 2.3
Connected modules Δ_0 and Δ_1.

Module Δ_0 sends two commsigs at commport α; independently, it may receive one commsig at commport β and one commsig at commport γ. Of course, comminst α_1 occurs before comminst α_2; this is the only order between comminsts of Δ_0. As a consequence, Δ_0 has twelve comminstorders, say ϕ_0 through ϕ_{11}:

$$\phi_0 \stackrel{\text{def}}{=} <\varnothing, \varnothing>$$
$$\phi_1 \stackrel{\text{def}}{=} <\{\beta_1\}, \varnothing>$$
$$\phi_2 \stackrel{\text{def}}{=} <\{\gamma_1\}, \varnothing>$$
$$\phi_3 \stackrel{\text{def}}{=} <\{\beta_1, \gamma_1\}, \varnothing>$$
$$\phi_4 \stackrel{\text{def}}{=} <\{\alpha_1\}, \varnothing>$$
$$\phi_5 \stackrel{\text{def}}{=} <\{\alpha_1, \beta_1\}, \varnothing>$$
$$\phi_6 \stackrel{\text{def}}{=} <\{\alpha_1, \gamma_1\}, \varnothing>$$
$$\phi_7 \stackrel{\text{def}}{=} <\{\alpha_1, \beta_1, \gamma_1\}, \varnothing>$$
$$\phi_8 \stackrel{\text{def}}{=} <\{\alpha_1, \alpha_2\}, \{\alpha_1 \sqsubset_{\phi_8} \alpha_2\}>$$
$$\phi_9 \stackrel{\text{def}}{=} <\{\alpha_1, \alpha_2, \beta_1\}, \{\alpha_1 \sqsubset_{\phi_9} \alpha_2\}>$$
$$\phi_{10} \stackrel{\text{def}}{=} <\{\alpha_1, \alpha_2, \gamma_1\}, \{\alpha_1 \sqsubset_{\phi_{10}} \alpha_2\}>$$
$$\phi_{11} \stackrel{\text{def}}{=} <\{\alpha_1, \alpha_2, \beta_1, \gamma_1\}, \{\alpha_1 \sqsubset_{\phi_{11}} \alpha_2\}>$$

Module Δ_1 may receive two commsigs at commport δ; thereafter it sends one commsig at commport ζ, after which it sends one commsig at commport η. Comminst δ_1 occurs before comminst δ_2; δ_2 occurs before ζ_1; ζ_1 occurs before η_1. As a consequence, Δ_1 has five comminstorders, say ϕ_{12} through ϕ_{16}:

$$\phi_{12} \stackrel{\text{def}}{=} <\varnothing, \varnothing>$$
$$\phi_{13} \stackrel{\text{def}}{=} <\{\delta_1\}, \varnothing>$$
$$\phi_{14} \stackrel{\text{def}}{=} <\{\delta_1, \delta_2\}, \{\delta_1 \sqsubset_{\phi_{14}} \delta_2\}>$$
$$\phi_{15} \stackrel{\text{def}}{=} <\{\delta_1, \delta_2, \zeta_1\}, \{\delta_1 \sqsubset_{\phi_{15}} \delta_2, \delta_2 \sqsubset_{\phi_{15}} \zeta_1\}>$$
$$\phi_{16} \stackrel{\text{def}}{=} <\{\delta_1, \delta_2, \zeta_1, \eta_1\}, \{\delta_1 \sqsubset_{\phi_{16}} \delta_2, \delta_2 \sqsubset_{\phi_{16}} \zeta_1, \delta_2 \sqsubset_{\phi_{16}} \eta_1\}>$$

There are twelve pairs of comminstorders that can be used to construct a commsigorder: (ϕ_0, ϕ_{12}), (ϕ_4, ϕ_{12}), (ϕ_4, ϕ_{13}), (ϕ_8, ϕ_{12}), (ϕ_8, ϕ_{13}), (ϕ_8, ϕ_{14}), (ϕ_8, ϕ_{15}), (ϕ_8, ϕ_{16}), (ϕ_9, ϕ_{15}), (ϕ_9, ϕ_{16}), (ϕ_{10}, ϕ_{16}), and (ϕ_{11}, ϕ_{16}). There is no partner for comminstorders ϕ_1, ϕ_2, ϕ_3, ϕ_5, ϕ_6, and ϕ_7, since Δ_1 will not send any commsig until it has received two commsigs. From definition 2.23, we infer that $\mathbf{REDOC}\,\phi_j = \phi_j$ for $0 \le j \le 15$; furthermore, we infer that:

$$\mathbf{REDOC}\,\phi_{16} = <\{\delta_1, \delta_2, \zeta_1, \eta_1\}, \{\delta_1 \sqsubset_{\phi_{16}} \delta_2, \delta_2 \sqsubset_{\phi_{16}} \zeta_1, \delta_2 \sqsubset_{\phi_{16}} \eta_1\}>$$

After the combining of the comminstorders and the renaming of the comminsts into commsigs, we are left with five commsigorders, say ξ_0 through ξ_4:

$$\xi_0 = <\varnothing, \varnothing>$$

$$\xi_1 = <\{(\alpha,1,\delta)\}, \varnothing>$$

$$\xi_2 = <\{(\alpha,1,\delta),(\alpha,2,\delta)\}, \{(\alpha,1,\delta) \sqsubset_{\xi_2} (\alpha,2,\delta)\}>$$

$$\xi_3 = <\{(\alpha,1,\delta),(\alpha,2,\delta),(\zeta,1,\beta)\}$$
$$,\{(\alpha,1,\delta) \sqsubset_{\xi_3} (\alpha,2,\delta),(\alpha,2,\delta) \sqsubset_{\xi_3} (\zeta,1,\beta)\}$$
$$>$$

$$\xi_4 = <\{(\alpha,1,\delta),(\alpha,2,\delta),(\zeta,1,\beta),(\eta,1,\gamma)\}$$
$$,\{(\alpha,1,\delta) \sqsubset_{\xi_4} (\alpha,2,\delta),(\alpha,2,\delta) \sqsubset_{\xi_4} (\zeta,1,\beta),(\alpha,2,\delta) \sqsubset_{\xi_4} (\eta,1,\gamma)\}$$
$$>$$

The pair of comminstorders (ϕ_0, ϕ_{12}) yields commsigorder ξ_0; the pairs (ϕ_4, ϕ_{12}) and (ϕ_4, ϕ_{13}) both yield ξ_1; the pairs (ϕ_8, ϕ_{12}), (ϕ_8, ϕ_{13}), and (ϕ_8, ϕ_{14}), all three yield ξ_2; the pairs (ϕ_8, ϕ_{15}) and (ϕ_9, ϕ_{15}), both yield ξ_3; the pairs (ϕ_8, ϕ_{16}), (ϕ_9, ϕ_{16}), (ϕ_{10}, ϕ_{16}), and (ϕ_{11}, ϕ_{16}), all four yield ξ_4. When we restrict these five commsigorders to all possible initial sets of commsigs, we find one additional commsigorder ξ_5 ($\xi_5 = \xi_4 \restriction \{(\alpha,1,\delta),(\alpha,2,\delta),(\eta,1,\gamma)\}$):

$$\xi_5 = <\{(\alpha,1,\delta),(\alpha,2,\delta),(\eta,1,\gamma)\}$$
$$,\{(\alpha,1,\delta) \sqsubset_{\xi_5} (\alpha,2,\delta),(\alpha,2,\delta) \sqsubset_{\xi_5} (\eta,1,\gamma)\}$$
$$>$$

We now have constructed interconnection Π_0 between modules Δ_0 and Δ_1 that are connected in the way described above: opdir $\mathbf{OP}\,\Pi_0$ is equal to $<\{\beta,\gamma\},\{\delta\}>$ and communication $\mathbf{CM}\,\Pi_0$ is equal to $\{\xi_0, \xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$.

end of example

When constructing the interconnection between two modules as desribed in this subsection, we are not concerned with the correctness concerns "absence of computation interference hazard" and "absence of transmission interference hazard". We deal with these when we address composition, see chapter 6.

2.1.7 Overview of interpretative issues

In table 2.4 we present the relation between our Communication Model and the underlying physics.

modeling communication	
Communication Model	the physical model
module	mechanism
one or more commports	terminal
directly connected commports	coinciding terminals
indirectly connected commports	terminals connected by a wire
comminst	individual instance of signal at terminal
modules have a direct connection	mechanisms exchange signals via coinciding terminals
modules have an indirect connection	mechanisms exchange signals via wires
comminstorder of module	order in which a mechanism may exchange signals
interconnection	all coinciding terminals of two mechanisms and all wires between these mechanisms
commsig	individual instance of signal that propagates between two terminals
commsigorder of interconnection	order in which signals may happen that are exchanged by two mechanisms

table 2.4

Relation between Communication Model and underlying physics.

Of course, to our Communication Model one can relate another physical model or some particular notion of 'physical reality' that one considers as the underlying physics; as a consequence, the entries in the right column will vary in accordance with the particular physical model or notion of physical reality that one wants to relate to our Communication Model.

2.1.8 Notational convention

When we need variables in our Communication Model we use Greek letters that are not in the Latin alphabet. We do not use ε or Ω, since we use them for other purposes in trace theory, see subsection 1.4.0; we do not use ω, since it has been used in the extension of trace theory with infinite traces.

Lower case letters near the beginning of the Greek alphabet are used to denote commports. Lower case letters near the middle of the Greek alphabet are used to denote comminsts and commsigs. Lower case letters near the end of the Greek alphabet are used to denote comminstorders and commsigorders. Capital Greek letters are used to denote modules, components, interconnections, channels, and sets of commports, comminsts, commsigs, comminstorders, or commsigorders. We will use Γ or Δ to denote a module or a component, Π to denote an interconnection, and Θ to denote a channel. We use Ψ to denote a set of commports, Λ to denote a set of comminsts or commsigs, Φ to denote an iodir or a set of comminstorders, and Ξ to denote an opdir or a set of commsigorders.

When we refer to specific objects, e.g. in examples, we use indexes. We use natural numbers as indexes to refer to specific objects locally, i.e. within one chapter of this monograph. When we want to refer to a specific object throughout the chapters of this monograph we use letters (or short words) as indexes.

As stated in subsection 1.4.4, boldface lower case operators are used in trace theory; boldface upper case operators are used in our Communication Model.

2.2 Introduction of trace theory in our Communication Model

In subsection 2.2.0 we associate notions in trace theory with commports, comminsts, and comminstorders; we associate notions in trace theory with commsigs and commsigorders in subsection 2.2.1. In subsection 2.2.2 we introduce the notions opdir and iodir in our Communication Model and we associate notions in trace theory with them. We abstract components from modules in subsection 2.2.3. In subsection 2.2.4 we abstract channels from interconnections. In subsection 2.2.5 we address the difference between our usage of the trace theory formalism and the earlier usage of directed trace structures to model delay-insensitive communication.

2.2.0 Commports, comminsts, and comminstorders

With commports and comminsts we associate *symbols*. With a commport α and each of its comminsts α_i the same symbol is associated. We associate the same symbol with either of two matching commports. With two commports that do not match, we associate distinct symbols.

With a comminstorder (strict partial order) we associate a *trace set*, viz. the set that consists of every full order (trace) that is consistent with the strict partial order (comminstorder). With every comminst in a comminstorder we associate a distinct symbol in every trace of the trace set that is associated with this comminstorder, see example 2.25.

example 2.25

Symbols *a* and *b* are associated with commports α and β, respectively. In trace *aba* we associate the leftmost occurrence of *a* with α_1, *b* with β_1, and the rightmost occurrence of *a* with α_2. With the comminstorder ϕ, in which

- comminst α_1 occurs before comminsts α_2 and β_1, and
- β_1 occurs before α_2,

viz. $\phi = <\{\alpha_1, \alpha_2, \beta_1\}, \{\alpha_1 \sqsubset_\phi \beta_1, \beta_1 \sqsubset_\phi \alpha_2\}>$, we associate trace set $\{aba\}$.

end of example

A trace is a totally ordered object (full order). Let symbols *a* and *b* be associated with commports α and β, respectively. The occurrence of comminst α_i before comminst β_j in a comminstorder is modeled by

> in every trace of the trace set that is associated with this comminstorder, the *i*-th occurrence of *a* is to the left of the *j*-th occurrence of *b*.

To model that comminsts α_i and β_j occur independently, we include in the trace set both: traces in which the *i*-th occurrence of *a* is to the left of the *j*-th occurrence of *b* and traces in which the *j*-th occurrence of *b* is to the left of the *i*-th occurrence of *a*.

example 2.26

Symbols *a* and *b* are associated with commports α and β, respectively. Trace set $\{ab\}$ is associated with the comminstorder ϕ in which comminst α_1 occurs before comminst β_1: $\phi = <\{\alpha_1, \beta_1\}, \{\alpha_1 \sqsubset_\phi \beta_1\}>$.

end of example

All traces in the trace set that is associated with a comminstorder have the same bag of symbols; as a consequence, they all have the same length. If and only if two comminsts are ordered in a comminstorder, the symbols, that are associated

with these two comminsts, occur in this same order in every trace of the trace set. On the other hand, if two comminsts are not ordered in a comminstorder, then there are traces (in the trace set that is associated with this comminstorder) in which the symbols, that are associated with these two comminsts, occur in one order, and there are traces (in this trace set) in which these symbols occur in the other order, see example 2.27.

example 2.27

Symbols a, b, and c are associated with commports α, β, and γ, respectively. With the comminstorder ϕ, in which

- comminst α_1 occurs before comminst β_1, and
- comminst γ_1 occurs independently of α_1 and β_1,

viz. $\phi = <\{\alpha_1, \beta_1, \gamma_1\}, \{\alpha_1 \sqsubset_\phi \beta_1\}>$, we associate trace set $\{abc, acb, cab\}$.

end of example

2.2.1 Commsigs and commsigorders

With commsigs we associate *symbols*. We consider output commport α and input commport β such that $\alpha \mathit{MATCH} \beta$. Let symbol a be associated with α and β, cf. subsection 2.2.0. We associate symbol a with every commsig (α, n, β), for $n \geq 1$.

Analogously to comminstorders, we associate with a commsigorder a *trace set*, viz. the set that consists of every full order (trace) that is consistent with the strict partial order (commsigorder). With every commsig in a commsigorder we associate a distinct symbol in every trace of the trace set that is associated with this commsigorder.

Again, all traces in the trace set that is associated with a commsigorder have the same bag of symbols; as a consequence, they all have the same length. If and only if two commsigs are ordered in a commsigorder, the symbols, that are associated with these two commsigs, occur in this same order in every trace of the trace set. On the other hand, if two commsigs are not ordered in a commsigorder, then there are traces (in the trace set that is associated with this commsigorder) in which the symbols, that are associated with these two commsigs, occur in one order, and there are traces (in this trace set) in which these symbols occur in the other order.

2.2.2 Opdirs and iodirs

An opdir consists of two disjoint sets of commports. In trace theory we associate an *alphbip* with an opdir. With opdir Ξ, alphbip $\mathbf{ab}\Xi$ is associated. The union of the sets of symbols that are associated with the two sets of commports of Ξ is called the *alphabet of* Ξ, which is denoted by $\mathbf{a}\Xi$. Of course, $\mathbf{a}\Xi = \mathbf{a}(\mathbf{ab}\Xi)$.

An *iodir* consists of a set of input commports and a set of output commports. In trace theory we associate an *iobip* with an iodir. With iodir Φ, the iobip $\mathbf{io}\Phi$ is associated. The set of symbols that are associated with the input commports of Φ is denoted by $\mathbf{i}\Phi$, the *input alphabet*; the set of symbols that are associated with the output commports of Φ is denoted by $\mathbf{o}\Phi$, the *output alphabet*.

property 2.28

For iodir Φ,

$$\mathbf{a}\Phi = \mathbf{o}\Phi \cup \mathbf{i}\Phi$$
$$\mathbf{i}(\mathbf{io}\Phi) = \mathbf{i}\Phi$$
$$\mathbf{o}(\mathbf{io}\Phi) = \mathbf{o}\Phi$$

end of property

From definition 2.11, reflection of iodir, we infer property 2.29.

property 2.29

For iodir Φ,

$$\mathbf{a}\bar{\Phi} = \mathbf{a}\Phi$$
$$\mathbf{o}\bar{\Phi} = \mathbf{i}\Phi$$
$$\mathbf{i}\bar{\Phi} = \mathbf{o}\Phi$$

end of property

2.2.3 Components

In subsection 2.0.3 we have introduced modules. The communication behavior of a module is a set of comminstorders. In subsection 2.2.0 we have associated a trace set with every comminstorder. As a consequence, a set of trace sets is associated with the set of comminstorders of a module. We define components as an abstraction from modules:

definition 2.30 *equivalence class of modules*

We call two modules equivalent if and only if they have

(i) the same iodir,

(ii) the same union of the trace sets that are associated with the comminstorders in the communication behavior of the module.

The equivalence classes are called *components*.

end of definition

Since the abstraction is confined to the communication behavior of modules, we feel free to discuss *commports*, *comminsts*, *sending, reception, and acceptance of commsigs*, and *open, closed, direct, indirect, and mixed connections* with respect to components as we do with respect to modules. Formally, a component Γ is a pair $<\mathbf{io}\,\Gamma, \mathbf{ptr}\,\Gamma>$. Iobip $\mathbf{io}\,\Gamma$ is called the *iobip of* Γ, and trace structure $\mathbf{ptr}\,\Gamma$ is called the *communication behavior of* Γ.

definition 2.31 *component*

We consider component Γ. Let Δ be a module in the equivalence class component Γ. Now Γ is defined by

(i) $\mathbf{io}\,\Gamma \stackrel{\text{def}}{=} \mathbf{io}(\mathbf{IO}\,\Delta)$

(ii) $\mathbf{a}(\mathbf{ptr}\,\Gamma) \stackrel{\text{def}}{=} \mathbf{a}(\mathbf{IO}\,\Delta)$

(iii) $\mathbf{t}(\mathbf{ptr}\,\Gamma)$ is defined as the union of all trace sets that are associated with the comminstorders of Δ.

end of definition

The definition of $\mathbf{t}(\mathbf{ptr}\,\Gamma)$ in definition 2.31(iii) is independent of the particular choice of the module in the equivalence class Γ, cf. definition 2.30(ii). Since for component Γ, $\mathbf{a}(\mathbf{ptr}\,\Gamma)=\mathbf{a}(\mathbf{io}\,\Gamma)$, we could have defined a component as an iobip – "trace set" pair in stead of as an iobip – "trace structure" pair. We have chosen not to do so, while on the one hand we like to separate the iobip from the communication behavior, and on the other hand trace sets are not very well suited for modeling composition due to the absence of associativity, cf. [Rem 85, Rem – van de Snepscheut – Udding 83].

In remark 3.16 we argue that we may lose some information when abstracting from module to component. In example 2.32 we present two distinct modules that are in one equivalence class. In the examples in this subsection 2.2.3 we use Δ to denote a module and Γ to denote a component.

example 2.32

A module Δ_2 has two output commports α and β and no input commports. Only one comminst of each commport can occur. They occur independently of each other.

$$\Psi^o_{\Delta_2} = \{\alpha, \beta\}$$
$$\Psi^o_{\Delta_2} = \varnothing$$
$$\mathbf{CB}\,\Delta_2 = \{<\varnothing,\varnothing>, <\{\alpha_l\},\varnothing>, <\{\beta_l\},\varnothing>, <\{\alpha_l, \beta_l\},\varnothing>\}$$

Let module Δ_2 be a member of the equivalence class component Γ_2. Let symbols *a* and *b* be associated with commports α and β, respectively:

$$\mathbf{o}(\mathbf{io}\,\Gamma_2) = \{a, b\}$$
$$\mathbf{i}(\mathbf{io}\,\Gamma_2) = \varnothing$$
$$\mathbf{ptr}\,\Gamma_2 = <\{a, b\}, \{\varepsilon, a, b, ab, ba\}>$$

We consider module Δ_3. Module Δ_3 has two output commports α and β and no input commports. Only one comminst of each commport can occur. Either α_l occurs before β_l or β_l occurs before α_l.

$$\Psi^o_{\Delta_3} = \{\alpha, \beta\}$$
$$\Psi^o_{\Delta_3} = \varnothing$$
$$\mathbf{CB}\,\Delta_3 = \{<\varnothing,\varnothing>, <\{\alpha_l\},\varnothing>, <\{\beta_l\},\varnothing>$$
$$, <\{\alpha_l, \beta_l\}, \{\alpha_l \sqsubset \beta_l\}>, <\{\alpha_l, \beta_l\}, \{\beta_l \sqsubset \alpha_l\}>$$
$$\}$$

Module Δ_3 also is a member of the equivalence class Γ_2.

end of example

The *alphabet of component* Γ is the set of symbols that are associated with the commports of Γ; it is denoted by $\mathbf{a}\Gamma$. The set of symbols that are associated with the output commports of component Γ is called the *output alphabet* of Γ, which is denoted by $\mathbf{o}\Gamma$. The set of symbols that are associated with the input commports of component Γ is called the *input alphabet* of Γ, which is denoted by $\mathbf{i}\Gamma$. We also associate an alphbip with Γ, which is denoted by $\mathbf{ab}\Gamma$. Even a symbol that is associated with a commport of a component at which no commsig will be received or sent, is an element of the alphabet of this component: the alphabet of a component is not restricted to symbols that occur in some trace in the trace structure of the component.

property 2.33

For component Γ,

(i) $\mathbf{a}\Gamma = \mathbf{a}(\mathbf{ptr}\,\Gamma)$

(ii) $\mathbf{a}\Gamma = \mathbf{o}\Gamma \cup \mathbf{i}\Gamma$

(iii) $\mathbf{o}\Gamma = \mathbf{o}(\mathbf{io}\,\Gamma)$

(iv) $\mathbf{i}\Gamma = \mathbf{i}(\mathbf{io}\,\Gamma)$

(v) $\mathbf{ab}\Gamma = \mathbf{o}\Gamma \oplus \mathbf{i}\Gamma$

end of property

Notice that $\oplus$ is a symmetric operator, see property 1.31.

Using postulate 2.12 we infer property 2.34.

property 2.34

For component Γ,

(i) $\varepsilon \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$

(ii) $\mathbf{ptr}\,\Gamma$ is prefix-closed

end of property

Of course, the input alphabet and the output alphabet of a component are not interchangeable, cf. example 2.35.

example 2.35

We consider components Γ_4 and Γ_5. Their output alphabets, input alphabets, and trace structures are defined by:

$$\mathbf{o}\Gamma_4 \stackrel{\text{def}}{=} \{a\} \qquad \mathbf{i}\Gamma_4 \stackrel{\text{def}}{=} \{b\}$$
$$\mathbf{o}\Gamma_5 \stackrel{\text{def}}{=} \{b\} \qquad \mathbf{i}\Gamma_5 \stackrel{\text{def}}{=} \{a\}$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma_4) \stackrel{\text{def}}{=} \{\varepsilon, a, ab\}$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma_5) \stackrel{\text{def}}{=} \mathbf{t}(\mathbf{ptr}\,\Gamma_4)$$

Components Γ_4 and Γ_5 differ: Γ_4 may initially send a commsig (comminst to which a is associated), after which it has to be able to accept a commsig (comminst to which b is associated); initially, Γ_5 has to be able to accept a commsig (comminst to which a is associated), after which it may send a commsig (comminst to which b is associated).

end of example

We extend definition 2.11, reflection of iodir, to components. The reflection of a component is a component.

definition 2.36 *reflection of component*

For component Γ, component $\overline{\Gamma}$ is the reflection of Γ; it is defined by

$$\mathbf{o}\overline{\Gamma} \stackrel{\text{def}}{=\!=} \mathbf{i}\Gamma$$
$$\mathbf{i}\overline{\Gamma} \stackrel{\text{def}}{=\!=} \mathbf{o}\Gamma$$
$$\mathbf{ptr}\,\overline{\Gamma} \stackrel{\text{def}}{=\!=} \mathbf{ptr}\,\Gamma$$

end of definition

In subsection 1.4.2 we have introduced state graphs to denote trace sets and trace structures. We also use a *state graph* to denote a component, say Γ; we shall only do this if every symbol of $\mathbf{a}\Gamma$ occurs in at least one trace of $\mathbf{t}(\mathbf{ptr}\,\Gamma)$. If $\mathbf{t}(\mathbf{ptr}\,\Gamma)$ has a regular state graph, then in the diagram of this state graph we shall postfix the symbols of $\mathbf{i}\Gamma$ with a question mark ($?$), and we shall postfix the symbols of $\mathbf{o}\Gamma$ with an exclamation mark ($!$); in figure 2.5 we show such a diagram, see example 2.37.

example 2.37

We consider the module that we presented in example 2.18. In this example we call it Δ_6. Module Δ_6 has one output commport γ and two input commports α and β. Only one comminst of each commport can occur. Comminsts α_l and β_l occur independently of each other; comminst γ_l occurs after both α_l and β_l have occurred.

$$\Psi^o_{\Delta_6} = \{\gamma\}$$
$$\Psi^i_{\Delta_6} = \{\alpha, \beta\}$$
$$\mathbf{CB}\,\Delta_6 = \{<\varnothing,\varnothing>, <\{\alpha_l\},\varnothing>, <\{\beta_l\},\varnothing>, <\{\alpha_l, \beta_l\},\varnothing> \\ , <\{\alpha_l, \beta_l, \gamma_l\}, \{\alpha_l \sqsubset \gamma_l, \beta_l \sqsubset \gamma_l\}> \\ \}$$

Let module Δ_6 be a member of the equivalence class component Γ_6. Let symbols a, b, and c be associated with commports α, β, and γ, respectively:

$$\mathbf{o}\Gamma_6 = \{c\}$$
$$\mathbf{i}\Gamma_6 = \{a, b\}$$
$$\mathbf{ptr}\,\Gamma_6 = <\{a, b, c\}, \{\varepsilon, a, b, ab, ba, abc, bac\}>$$

The state graph of Γ_6 is shown in figure 2.5.

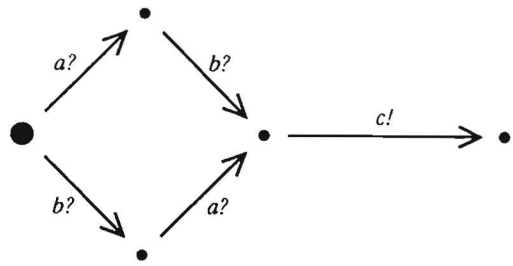

figure 2.5
State graph of component Γ_6.

end of example

example 2.38

A module Δ_7 has one output commport α and one input commport β. Output comminst α_1 occurs before input comminst β_1; output comminst α_2 occurs after input comminst β_1.

$$\Psi^o_{\Delta_7} = \{\alpha\}$$
$$\Psi^i_{\Delta_7} = \{\beta\}$$
$$\mathbf{CB}\,\Delta_7 = \{<\varnothing,\varnothing>, <\{\alpha_1\},\varnothing>, <\{\alpha_1,\beta_1\},\{\alpha_1 \sqsubset \beta_1\}>$$
$$, <\{\alpha_1,\beta_1,\alpha_2\},\{\alpha_1 \sqsubset \beta_1, \beta_1 \sqsubset \alpha_2\}>$$
$$\}$$

Let module Δ_7 be a member of the equivalence class component Γ_7. Let symbols a and b be associated with commports α and β, respectively:

$$\mathbf{o}\,\Gamma_7 = \{a\}$$
$$\mathbf{i}\,\Gamma_7 = \{b\}$$
$$\mathbf{ptr}\,\Gamma_7 = <\{a,b\}, \{\varepsilon, a, ab, aba\}>$$

We notice that the two comminsts α_1 and α_2 are explicitly distinguished from each other by their indexes. In the traces of trace set $\mathbf{ptr}\,\Gamma_7$, however, this explicit distinction is not present.

end of example

2.2.3.0 Enabling and disabling

We consider a component Γ. Γ has two commports: α and β. With these commports we associate symbols a and b, respectively. We say that comminst α_i *enables* comminst β_j in Γ (for $i \geq 1$ and $j \geq 1$) if and only if $(\mathbf{E}\, t : t \in \Omega^* \wedge (\#_a t = i-1) \wedge (\#_b t = j-1) : tb \notin \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge tab \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$. We give an example of the enable relation in example 2.39.

Analogously, we say that comminst α_i *disables* comminst β_j in Γ if and only if $(\mathbf{E}\, t : t \in \Omega^* \wedge (\#_a t = i-1) \wedge (\#_b t = j-1) : tb \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge tab \notin \mathbf{t}(\mathbf{ptr}\,\Gamma))$. We notice that every comminst disables itself. In example 2.39 we also give an example of the disable relation.

example 2.39

We consider component Γ_8. Γ_8 has one input commport α and two output commports β and γ. With commports α, β, and γ we associate symbols a, b, and c, respectively, see figure 2.6.

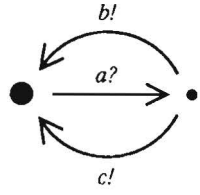

figure 2.6
State graph of component Γ_8.

In component Γ_8 comminst α_1 enables comminst β_1, since $b \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_8)$ and $ab \in \mathbf{t}(\mathbf{ptr}\,\Gamma_8)$. Analogously, α_1 enables γ_1 in Γ_8. In general, comminst α_k enables comminsts β_j and γ_j in component Γ_8 and β_j and γ_j both enable α_{k+1} in Γ_8 (for $1 \leq j \leq k$).

In component Γ_8 comminst β_1 disables comminst γ_1, since $ac \in \mathbf{t}(\mathbf{ptr}\,\Gamma_8)$ and $abc \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_8)$. Analogously, γ_1 disables β_1 in Γ_8. In general, comminst β_j disables comminst γ_k in component Γ_8 and γ_k disables β_j in Γ_8 (for $j \geq 1$ and $k \geq 1$); furthermore, every comminsts disables itself in Γ_8.

end of example

We could have defined more sophisticated "enabling" and "disabling" relations, e.g. on triples of two comminsts and a comminstorder. Since we do not need such sophisticated relations, we didn't choose to define them in this monograph.

remark 2.40

Notice that the "enabling" and "disabling" relations do not exclude each other: it is possible that comminst α_i enables and disables comminst β_j in Γ (for $i \geq 1$ and $j \geq 1$), see example 2.41.

end of remark

example 2.41

We consider component Γ_9. Γ_9 has no input commports and three output commports α, β, and γ. With commports α, β, and γ we associate symbols a, b, and c, respectively, see figure 2.7.

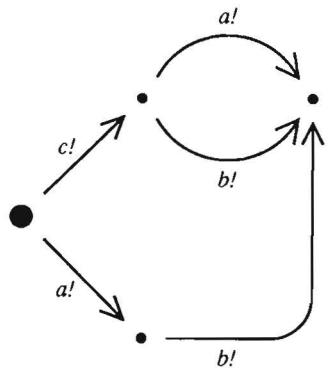

figure 2.7
State graph of component Γ_9.

In component Γ_9 comminst α_1 enables comminst β_1, since $b \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_9)$ and $ab \in \mathbf{t}(\mathbf{ptr}\,\Gamma_9)$. Furthermore, α_1 disables β_1, since $cb \in \mathbf{t}(\mathbf{ptr}\,\Gamma_9)$ and $cab \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_9)$.

end of example

2.2.4 Channels

In this subsection we define channels as an abstraction from interconnections. This abstraction is analogous to the abstraction from modules to components in subsection 2.2.3.

definition 2.42 *equivalence class of interconnections*

We call two interconnections equivalent if and only if they have

(i) the same opdir,

(ii) the same union of the trace sets that are associated with the commsigorders in the communication of the interconnection.

The equivalence classes are called *channels*.

end of definition

Since the abstraction is confined to the communication of interconnections, we feel free to discuss *commports* and *commsigs* with respect to channels as we do with respect to interconnections. Formally, a channel Θ is a pair $<\mathbf{ab}\Theta, \mathbf{ptr}\,\Theta>$. Alphbip $\mathbf{ab}\Theta$ is called the *alphbip of* Θ, and trace structure $\mathbf{ptr}\,\Theta$ is called the *communication of* Θ.

definition 2.43 *channel*

We consider channel Θ. Let Π be an interconnection in the equivalence class channel Θ.

(i) $\mathbf{ab}(\mathbf{ptr}\,\Theta) \stackrel{\text{def}}{=} \mathbf{ab}(\mathbf{OP}\,\Pi)$

(ii) $\mathbf{a}(\mathbf{ptr}\,\Theta) \stackrel{\text{def}}{=} \mathbf{a}(\mathbf{OP}\,\Pi)$

(iii) $\mathbf{t}(\mathbf{ptr}\,\Theta)$ is defined as the union of all trace sets that are associated with the commsigorders of Π.

end of definition

The definition of $\mathbf{t}(\mathbf{ptr}\,\Theta)$ in definition 2.43(iii) is independent of the particular choice of the module in the equivalence class Θ, cf. definition 2.42(ii).

The *alphabet of channel* Θ is the set of symbols that are associated with the commports of channel Θ; it is denoted by $\mathbf{a}\Theta$. Of course, $\mathbf{a}\Theta = \mathbf{a}(\mathbf{ptr}\,\Theta)$. Like the alphabet of a component, the alphabet of a channel is not restricted to symbols that occur in some trace in the trace structure of the channel.

property 2.44

For channel Θ and sets of symbols A and B such that $\mathbf{ab}\Theta = A \oplus B$,

$$\mathbf{a}\Theta = A \cup B$$

end of property

From postulate 2.14 we infer property 2.45.

property 2.45

For channel Θ,

(i) $\varepsilon \in \mathbf{t}(\mathbf{ptr}\,\Theta)$

(ii) $\mathbf{ptr}\,\Theta$ is prefix-closed

end of property

property 2.46 *alphbip of channel between two components*

For channel Θ between components Γ and Δ,

$$\mathbf{ab}\Theta = (\mathbf{o}\Gamma \cap \mathbf{i}\Delta) \oplus (\mathbf{o}\Delta \cap \mathbf{i}\Gamma).$$

end of property

2.2.5 Comparison with the use of directed trace structures

Until now directed trace structures have been used to model delay-insensitive communication, cf. [van de Snepscheut85, Udding84, Schols85, Verhoeff85, Black86, Ebergen87, Schols88, Dill88]. In directed trace structures the alphabet is partitioned into disjoint, possibly empty sets, for example the "input alphabet" and the "output alphabet".

In this monograph we use (undirected) trace structures to model either the communication in an interconncetion or the communication behavior of a module.

We consider directions to be issues that are related to the use of an interconnection or to the use of the commports by a module. Hence, directions are interpretative issues. For this reason we use (undirected) trace structures to model the communication of an interconnection (channel) and the communication behavior of a module (component).

The use of (undirected) trace structures in this monograph leads to formally different definitions of properties such as delay-safety, delay-insensitivity, computation interference hazard, and transmission interference hazard. These now appear as properties of channels and/or components, see chapters 4 and 5. The redefinitions given here have equivalent consequences as the definitions given earlier, see [Udding84, Schols85, Verhoeff85, Ebergen87].

2.3 Examples of components

In this section we give some examples of components. These components will be used in the following chapters.

example 2.47 *"Wire" element*

We consider a mechanism that is a "Wire" element. It has one input and one output. Initially the input is low, the output is low, and there are no signals on their way. When the input is high the output may go high; when the input is low the output may go low. No input change is allowed whenever an output change is pending. The transitions between low and high (and vice versa) are modeled as the comminsts of component Γ_w, see figure 2.8.

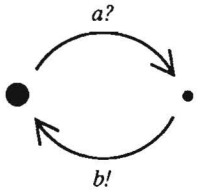

figure 2.8
State graph of component Γ_w.

Symbol a is associated with the input commport of Γ_w; symbol b is associated with the output commport.

We consider the same mechanism with a different initial condition: initially the input is high, the output is low, and there is one signal on its way. We call the mechanism with this initial condition a "Wire with Initial Transition" element. The transitions between low and high (and vice versa) are modeled as the comminsts of component Γ_{wit}, see figure 2.9.

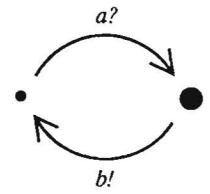

figure 2.9
State graph of component Γ_{wit}.

Symbol a is associated with the input commport of Γ_{wit}; symbol b is associated with the output commport.

end of example

example 2.48 *"Muller-C" element*

We consider a mechanism that is a "Muller-C" element. The "Muller-C" element is also called a "Rendez Vous" element. It has two inputs and one output. Initially both inputs are low and the output is low. When both inputs are high the output may go high; when both inputs are low the output may go low. No input that has the other value (low versus high) than the output, is allowed to change. The transitions between low and high (and vice versa) are modeled as the comminsts of component Γ_c, see figure 2.10.

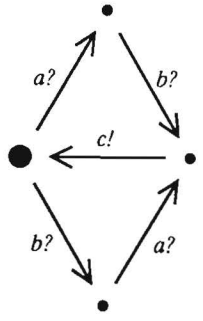

figure 2.10
State graph of component Γ_c.

Symbols a and b are associated with the input commports of Γ_c; symbol c is associated with the output commport.

end of example

example 2.49 *"Fork" elements*

We consider a mechanism that is a "Fork" element. It has one input and two outputs. Initially the input is low, both outputs are low, and there are no signals on their way. When the input is high the outputs may go high; when the input is low the outputs may go low. No input change is allowed whenever an output change is pending. The transitions between low and high (and vice versa) are modeled as the comminsts of component Γ_f, see figure 2.11.

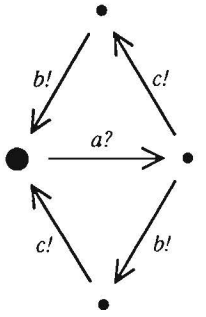

figure 2.11
State graph of component Γ_f.

Symbol a is associated with the input commport of Γ_f; symbols b and c are associated with the output commports.

We consider a mechanism that is an "Asymmetric Fork" element, see the scheme in figure 2.12.

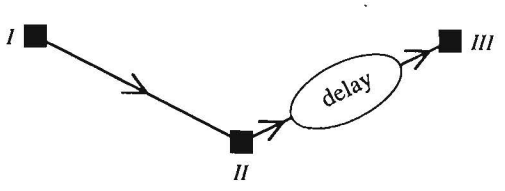

figure 2.12
Scheme of an Asymmetric Fork element.

The delay element has a delay that is large enough to guarantee that, after a signal has happened at the input terminal (I), a signal happens at the lower left output terminal (II) before a signal happens at the upper right output terminal (III). This mechanism is modeled by component Γ_{af}. Component Γ_{af} has input commport α and output commports β and γ, see figure 2.13.

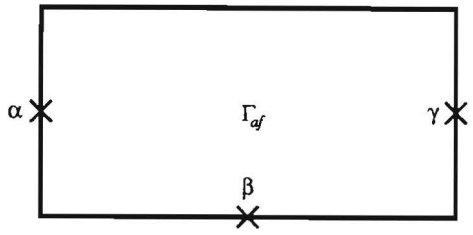

figure 2.13
Component Γ_{af}.

Output commport γ models the output terminal 'after the delay element'; output commport β models the other output terminal. We associate symbol *a* with input commport α of Γ_{af}; symbols *b* and *c* are associated with the output commports β and γ of Γ_{af}, respectively.

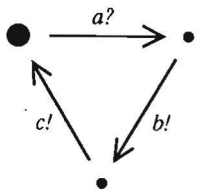

figure 2.14
State graph of component Γ_{af}.

The state graph of component Γ_{af} is shown in figure 2.14.
end of example

In example 2.50 we illustrate that a given mechanism might be associated with different behavioral abstractions. In our Communication Model this leads to –possibly different– components that model these different behavioral abstractions of a mechanism.

example 2.50 *wires with bundling constraint*

We consider a communication mechanism for which the delay from input to output is less for the data wires than for the control wire. There is one control wire and there are a number of data wires. The control wire is a so-called "data-valid wire". No input change on a wire is allowed whenever an output change on this wire is pending. Initially all inputs are low, all outputs are low, and there are no signals on their way. When a signal at the input of a data wire is received (by the communication mechanism) before a signal at the input of the control wire is received, this communication mechanism behaves such as to produce a signal at the output of the particular data wire before it produces a signal at the output of the control wire. In figure 2.15 we present a scheme of such a mechanism with one data wire.

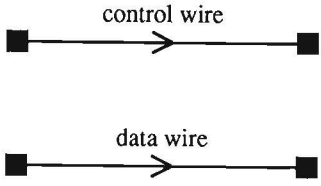

figure 2.15
Scheme of a mechanism that has a bundling constraint.

The most general use of such a mechanism with one data wire is modeled by component Γ_{bc}. The terminals of the control wire are modeled by input commport α and output commport β of Γ_{bc}; the terminals of the data wire are modeled by input commport γ and output commport δ of Γ_{bc}, see figure 2.16.

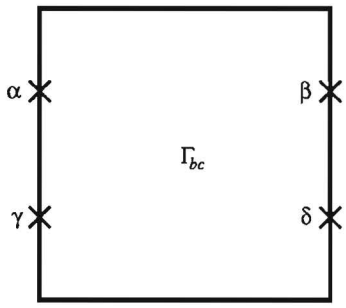

figure 2.16
Component Γ_{bc}.

We associate symbols *a* and *c* with input commports α and γ, respectively; we associate symbols *b* and *d* with output commports β and δ, respectively. The state graph of component Γ_{bc} is presented in figure 2.17.

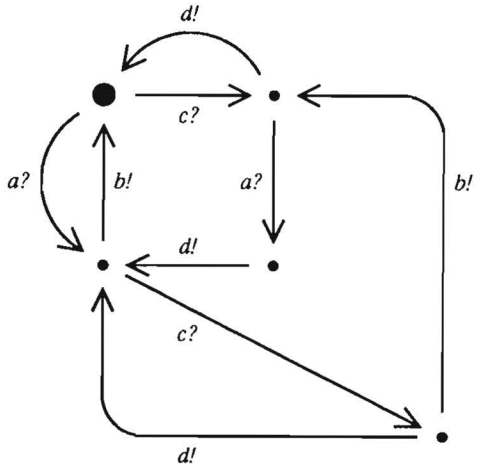

figure 2.17
State graph of component Γ_{bc}.

A signal at the input of the control wire of this mechanism may be received before a signal at the input of a data wire is received, i.e. the control and data wires can be used as normal wires. Still this communication mechanism differs from two normal wires due to the existing bundling constraint, cf. component Γ_{2w}, see figure 2.18.

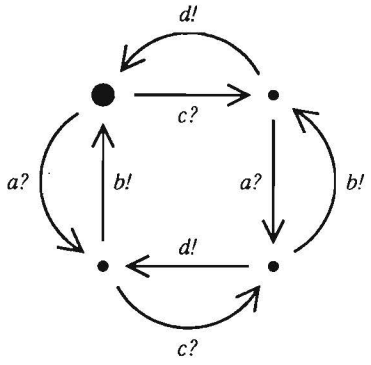

figure 2.18
State graph of component Γ_{2w}.

Component Γ_{bc} does not model the typical use of this mechanism. This use is modeled by component Γ_{1ubc}, see figure 2.19.

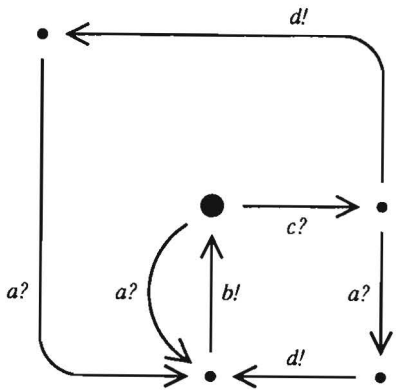

figure 2.19
State graph of component Γ_{lubc}.

The typical use modeled by Γ_{lubc} is a restriction of the general use as modeled by Γ_{bc}; formally, $\mathbf{ptr}\,\Gamma_{lubc} \subseteq \mathbf{ptr}\,\Gamma_{bc}$.

end of example

example 2.51 *"Or" and "And" elements*

We consider a mechanism that is an "Or" element. It has two inputs and one output. Initially both inputs are low, the output is low, and there are no signals on their way. When at least one of the inputs is high the output may go high; when both inputs are low the output may go low. There are no restrictions on input changes. The transitions between low and high (and vice versa) are modeled as the comminsts of component Γ_{or}, see figure 2.20.

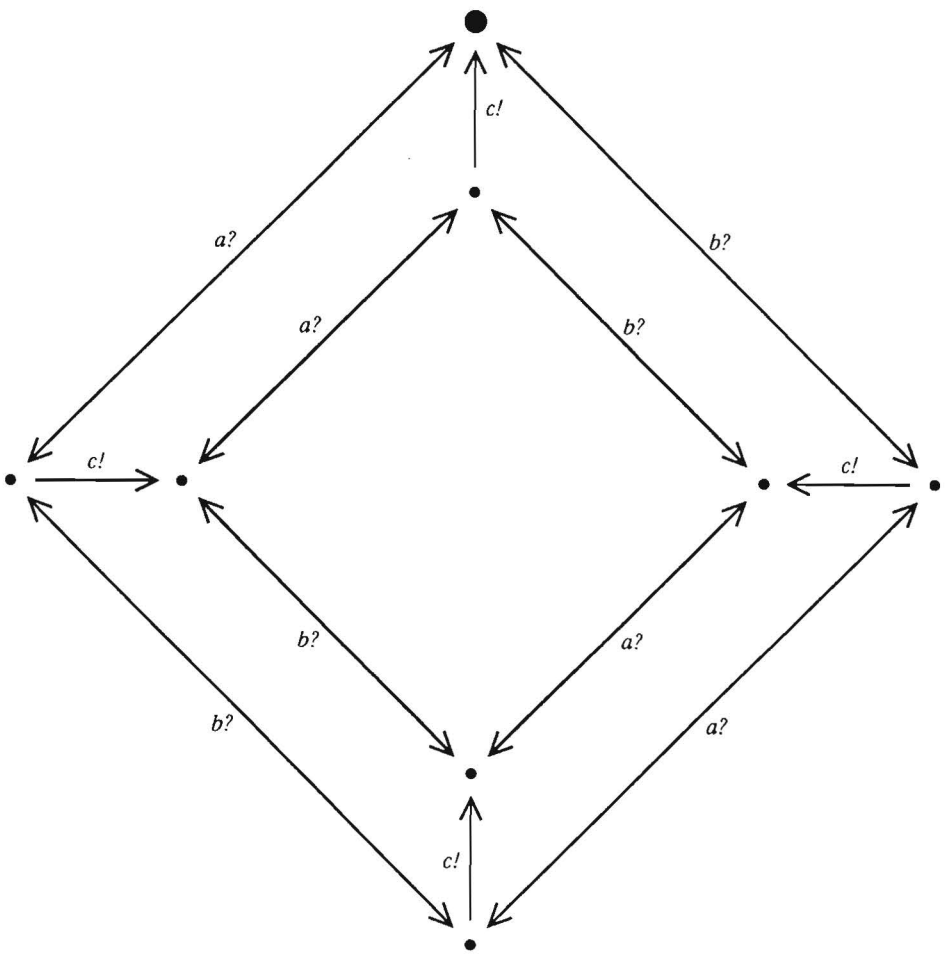

figure 2.20
State graph of component Γ_{or}.

Symbols a and b are associated with the input commports of Γ_{or}; symbol c is associated with the output commport.

Since our Communication Model is event-based, the only difference between component Γ_{or} and Γ_{and}, which models the "And" element, is the initial state, see figure 2.21.

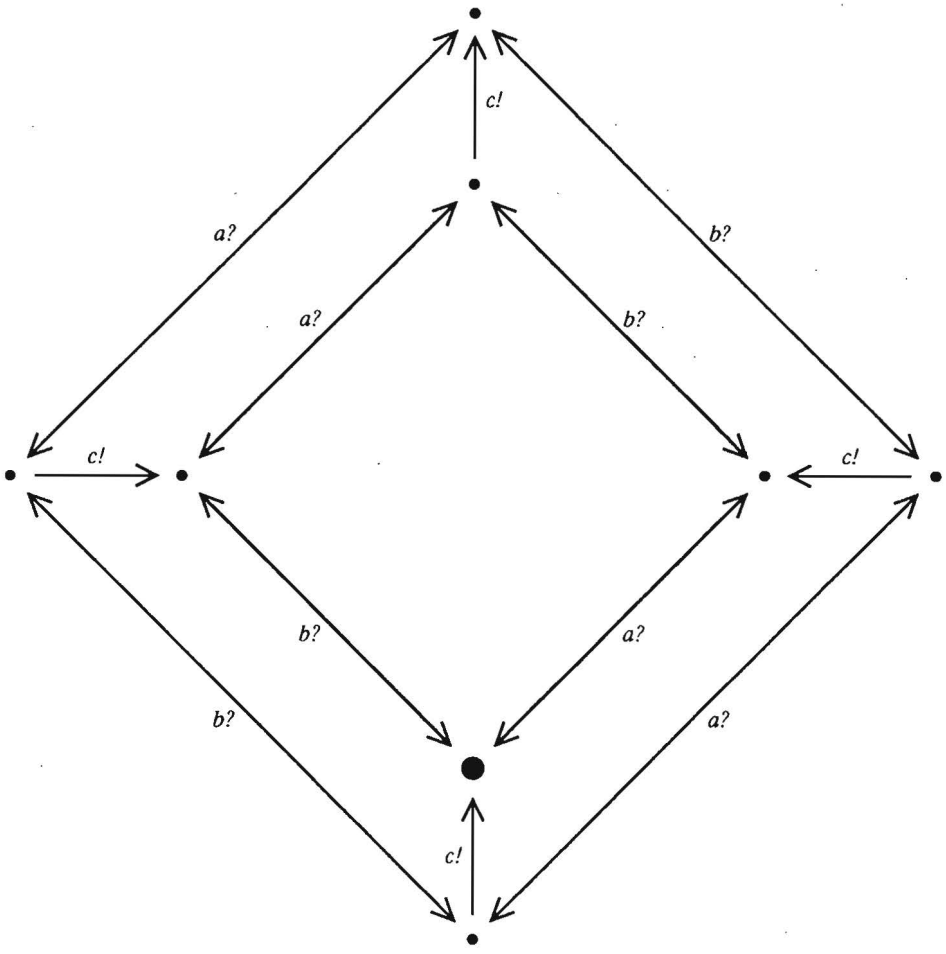

figure 2.21
State graph of component Γ_{and}.

end of example

example 2.52 *"Majority" element*

We consider a mechanism that is a "Majority" element. It has three inputs and one output. Initially all inputs are low and there are no signals on their way. When at least two of the inputs are high the output may go high; when at least two of the inputs are low the output may go low. There are no restrictions on input changes. The transitions between low and high (and vice versa) are modeled as the comminsts of component Γ_{maj}, see figure 2.22.

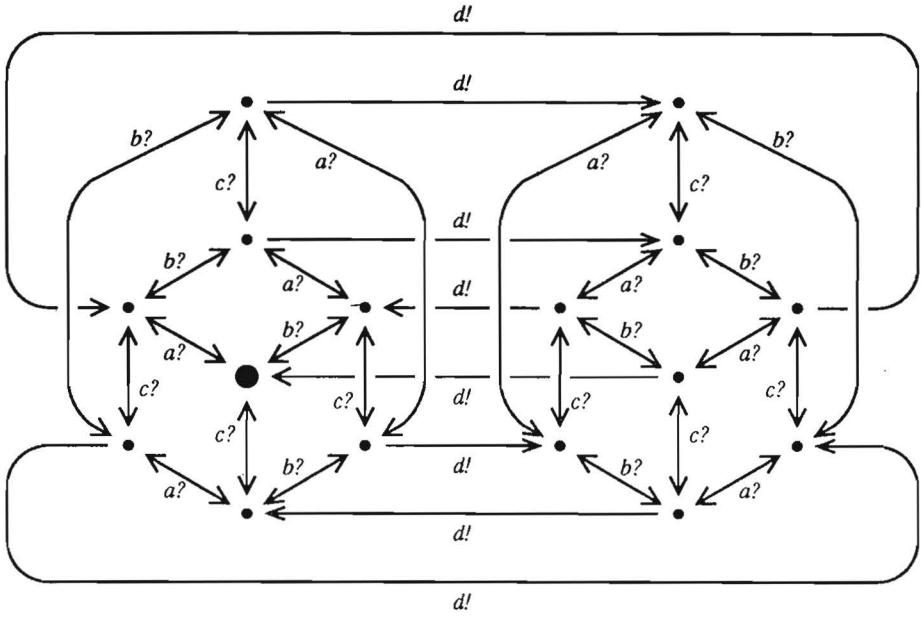

figure 2.22
State graph of component Γ_{maj}.

Symbols a, b, and c are associated with the input commports of Γ_{maj}; symbol d is associated with the output commport.

The diagram of the state graph in figure 2.22 is not minimal. Nevertheless, we have chosen to show this diagram. The reason for doing so is clearness.

end of example

2.4 Event-based model

We have chosen to make our Communication Model an *event-based* model. In an event-based model the changes of inputs and outputs are modeled. In a state-based model the states of the inputs and outputs, e.g. low or high, are modeled. We prefer an event-based model to a state-based model, see also [Rem91], because we want to model delay-safe and delay-insensitive communication: there are no clocks in our model and there is no sampling of 'states of terminals (or wires)'. Notice that trace theory also is an event-based model, see [Rem – van de Snepscheut – Udding83].

States and state graphs are derived notions in our Communication Model. A state is an equivalence class of traces; our states do, in general, not correspond to the 'states of wires', cf. example 2.53.

example 2.53

Component Γ_{10} is given by the state graph in figure 2.23.

figure 2.23
State graph of component Γ_{10}.

We see that symbol a is associated with the output commport of Γ_{10} and symbol b is associated with the input commport of Γ_{10}. Let us assume that Γ_{10} models a mechanism of which the terminals are connected to wires; and let us assume that initially these wires are low (i.e. they have a voltage that corresponds to the logical 0), and that transitions change the wires from low to high and vice versa. We see that in the state graph of Γ_{10} in our Communication Model the state that contains trace ε differs from the state that contains trace *abab*; the 'state of both wires' in the physical model, however, is equal to low in both cases. This is why in a "four-phase handshake protocol" extra variables are needed, cf. [Martin85b].

On the other hand we consider component Γ_w, see example 2.47. The initial state in the state graph of Γ_w contains trace *ab*. However, in the mechanism the 'state of the terminals' is initially low, whereas after the comminsts with which a and b are associated have occurred, the 'state of the terminals' is high.

end of example

Notice that we do not assume that the mechanisms, which implement the components in the physical model, are designed event-driven: we have only chosen to model the communication in our Communication Model event-based. For a detailed treatment of the event-driven (transition-signaling) concept we refer to [Seitz80]. For an example of the transition-signaling conceptual framework we refer to the micropipelines in [Sutherland89].

3

Computation interference hazard

In this chapter we define the correctness concern *absence of computation interference hazard*. Furthermore, we present a technique to 'transform' other correctness concerns into absence of computation interference hazard. An example of such an other correctness concern is "absence of transmission interference hazard". Josephs and Udding have chosen an opposite approach: they 'transform' absence of computation interference hazard into absence of transmission interference hazard, see [Josephs – Udding 90].

In this chapter we study the communication between two components that have a closed direct connection. At some places we refer to one component only; then the environment of this component implicitly plays the role of the other component. In section 3.0 we explain why we often refer to "computation interference hazard" when we discuss the phenomenon "computation interference". Computation interference hazard can arise when we compose components. In order to compose components, we have to connect them in a proper way. This is discussed in section 3.1. In section 3.2 we define computation interference hazard formally. In the next chapters we will transform some phenomenon hazards into computation interference hazard. The general transformation technique is presented in section 3.3.

3.0 Hazards

In our Communication Model we shall refer to some (undesired) phenomena, viz. "computation interference", "transmission interference", and "ambiguous quiescence", using the word *hazard* in order to indicate that it is possible for such a phenomenon to occur. The "phenomenon hazard" is a weaker notion than a guaranteed occurrence of the phenomenon. As a consequence, given some phenomenon, "absence of phenomenon hazard" is a stronger notion than "absence of (any guaranteed occurrence of) this phenomenon": when we have proven "absence of phenomenon hazard", we may conclude that the phenomenon is not present. The name hazard originates from switching theory, cf. [Unger69], where it has the same connotation that we attach to it now.

There exists "computation interference" if a mechanism receives a signal that it doesn't accept, cf. subsection 2.1.3. We say that there exists "computation interference hazard" if we cannot guarantee that a mechanism only receives signals that it does accept.

example 3.0 *computation interference hazard*

We consider components Γ_0 and Δ_0. Γ_0 and Δ_0 have a direct connection. Γ_0 has one output commport (α) and one input commport (β); Δ_0 has one output commport (δ) and one input commport (γ). Commport α matches commport γ and commport δ matches commport β, see figure 3.74.

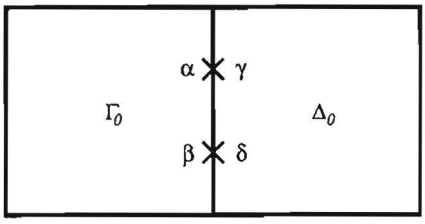

figure 3.0
Directly connected components Γ_0 and Δ_0.

We associate symbol *a* with commports α and γ; we associate symbol *b* with commports β and δ. Components Γ_0 and Δ_0 are defined by:

$$\mathbf{o}\Gamma_0 \stackrel{\text{def}}{=} \{a\} \qquad \mathbf{i}\Gamma_0 \stackrel{\text{def}}{=} \{b\} \qquad \mathbf{t}(\mathbf{ptr}\,\Gamma_0) \stackrel{\text{def}}{=} \{\varepsilon, a, ab\}$$
$$\mathbf{o}\Delta_0 \stackrel{\text{def}}{=} \{b\} \qquad \mathbf{i}\Delta_0 \stackrel{\text{def}}{=} \{a\} \qquad \mathbf{t}(\mathbf{ptr}\,\Delta_0) \stackrel{\text{def}}{=} \{\varepsilon, a, b, ab, ba\}$$

Initially, Δ_0 may send δ_1. In this case Γ_0 receives β_1 before it has sent α_1. This is not allowed according to $\mathbf{ptr}\,\Gamma_0$: Γ_0 does not accept β_1 before it has sent α_1. Thus there is an occurrence of "computation interference". If Δ_0 sends δ_1 after it has received γ_1, Γ_0 receives β_1 after it has sent α_1. In this case, there is no occurrence of computation interference.

Since Δ_0 may send δ_1 before it has received γ_1, it is possible that β_1 is received by Γ_0 before Γ_0 has sent α_1. Thus we cannot guarantee that there is no occurrence of computation interference. We say that there exists "computation interference hazard".

end of example

There exists "transmission interference" if two signals exchanged by two mechanisms interfere. We say that there exists "transmission interference hazard" if we cannot guarantee that two signals exchanged by two mechanisms do not interfere.

example 3.1 *transmission interference hazard*

We consider indirectly connected components Γ_I and Δ_I. Γ_I has one output commport (α); Δ_I has one input commport (β). Commport α matches commport β, see figure 3.1.

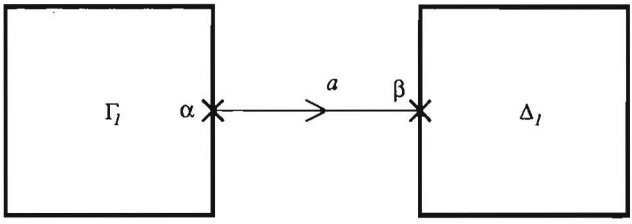

figure 3.1
Indirectly connected components Γ_I and Δ_I.

We associate symbol a with both commports. The components Γ_I and Δ_I are defined by:

$$\mathbf{o}\Gamma_I \stackrel{\text{def}}{=} \{a\} \qquad \mathbf{i}\Gamma_I \stackrel{\text{def}}{=} \varnothing \qquad \mathbf{t}(\mathbf{ptr}\,\Gamma_I) \stackrel{\text{def}}{=} \{\varepsilon, a, aa\}$$
$$\mathbf{o}\Delta_I \stackrel{\text{def}}{=} \varnothing \qquad \mathbf{i}\Delta_I \stackrel{\text{def}}{=} \{a\} \qquad \mathbf{t}(\mathbf{ptr}\,\Delta_I) \stackrel{\text{def}}{=} \{\varepsilon, a, aa\}$$

The mechanisms modeled by these components agree about the communication between them: the mechanism modeled by Γ_I sends two signals, which are accepted by the mechanism modeled by Δ_I. If the two signals interfere, there is an occurrence of "transmission interference". Since Γ_I may send α_2 before Δ_I has received β_1, it is possible that the two signals interfere. We say that there exists "transmission interference hazard".

end of example

3.1 Connected components

We associate the same symbol with either of two matching commports. The connection of the components must be such that no input commports are connected to each other and no output commports are connected to each other. This restriction is captured in the definition of "i/o-connectable".

definition 3.2 *i/o-connectable*

Components Γ and Δ are *i/o-connectable* if and only if

$$\mathbf{a}\Gamma \cap \mathbf{a}\Delta = (\mathbf{o}\Gamma \cap \mathbf{i}\Delta) \cup (\mathbf{i}\Gamma \cap \mathbf{o}\Delta)$$

end of definition

From definition 3.2, "i/o-connectable", we infer that i/o-connectable is a symmetric relation. The following property shows a different characterization of i/o-connectable.

property 3.3

Components Γ and Δ are i/o-connectable if and only if

$$(\mathbf{i}\Gamma \cap \mathbf{i}\Delta = \varnothing) \wedge (\mathbf{o}\Gamma \cap \mathbf{o}\Delta = \varnothing)$$

end of property

We notice that definition 3.2, "i/o-connectable", doesn't require that Γ and Δ have a closed connection. Nevertheless, in this chapter we are concerned with closed connections only. In chapter 6 we consider open connections.

3.2 Absence of computation interference hazard

In subsection 3.2.0 we present the definitions of "absence of computation interference hazard" for two components that have a closed direct connection. In subsection 3.2.1 we relate the acceptance of commsigs by a component to (absence of) computation interference.

remark 3.4

Absence of computation interference hazard is the correctness concern that has a central part in this monograph. Whatever we do, we always see to it that there is absence of computation interference hazard. This means that, whatever we do, we always establish that no commsig might be received by any component that is not able to accept this commsig.

end of remark

3.2.0 Direct connection

In this subsection we will define absence of computation interference hazard for two i/o-connectable components that have a closed direct connection. We first define absence of computation interference hazard *at one component* for two i/o-connectable components that have a closed direct connection. For all definitions of absence of computation interference hazard in this chapter we need two components. In chapter 6 we refer to absence of computation interference hazard in connections of more than two components.

definition 3.5 ***NCIHA***

For i/o-connectable components Γ and Δ, we define predicate Γ ***NCIHA*** Δ by:

$$\Gamma \textit{ NCIHA } \Delta \stackrel{\text{def}}{=} (\mathbf{A}\, t, u, a : ta \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge u \in \mathbf{t}(\mathbf{ptr}\,\Delta) \wedge a \in (\mathbf{o}\Gamma \cap \mathbf{i}\Delta) \wedge (t \restriction (\mathbf{a}\Gamma \cap \mathbf{a}\Delta) = u \restriction (\mathbf{a}\Gamma \cap \mathbf{a}\Delta)) : ua \in \mathbf{t}(\mathbf{ptr}\,\Delta))$$

end of definition

The predicate Γ ***NCIHA*** Δ, is the formalization of *there is absence of computation interference hazard at Δ for components Γ and Δ that have a direct connection*. If Γ ***NCIHA*** Δ, then Δ accepts every commsig that it may receive from Γ. To illustrate definition 3.5, "***NCIHA***", we present some examples.

example 3.6

We consider components Γ_2 and Γ_{wit}; they are defined by:

$$\mathbf{o}\Gamma_2 \stackrel{\text{def}}{=} \{a\} \quad \mathbf{i}\Gamma_2 \stackrel{\text{def}}{=} \{b\} \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_2) \stackrel{\text{def}}{=} \mathbf{pref}(\{ab\}^*)$$
$$\mathbf{o}\Gamma_{wit} = \{b\} \quad \mathbf{i}\Gamma_{wit} = \{a\} \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_{wit}) = \mathbf{pref}(\{ba\}^*)$$

Both components model a "Wire with Initial Transition" element, cf. example 2.47. Since $a \in \mathbf{t}(\mathbf{ptr}\,\Gamma_2)$, $\varepsilon \in \mathbf{t}(\mathbf{ptr}\,\Gamma_{wit})$, and $a \in (\mathbf{o}\Gamma_2 \cap \mathbf{i}\Gamma_{wit})$, but $a \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_{wit})$, we conclude from definition 3.5, "***NCIHA***", that $\neg(\Gamma_2$ ***NCIHA*** $\Gamma_{wit})$: there is computation interference hazard at Γ_{wit} for components Γ_2 and Γ_{wit} that have a direct connection.

By symmetry we conclude also that $\neg(\Gamma_{wit}$ ***NCIHA*** $\Gamma_2)$.

end of example

example 3.7

We consider components Γ_{wit}, cf. example 2.47, and Γ_3; they are given by:

$$\mathbf{o}\Gamma_{wit} = \{b\} \quad \mathbf{i}\Gamma_{wit} = \{a\} \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_{wit}) = \mathbf{pref}(\{ba\}^*)$$
$$\mathbf{o}\Gamma_3 \stackrel{\text{def}}{=} \{a\} \quad \mathbf{i}\Gamma_3 \stackrel{\text{def}}{=} \{b\} \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_3) \stackrel{\text{def}}{=} \mathbf{pref}(\{ab, ba\}^*)$$

We notice that Γ_{wit} ***NCIHA*** Γ_3, but $\neg(\Gamma_3$ ***NCIHA*** $\Gamma_{wit})$, cf. example 3.6.

end of example

Using definition 3.5, "***NCIHA***", we define absence of computation interference hazard for two components that have a direct connection.

definition 3.8 ***NCIH***

For i/o-connectable components Γ and Δ, we define predicate Γ ***NCIH*** Δ by:

$$\Gamma\ \textit{NCIH}\ \Delta \stackrel{\text{def}}{=} (\Gamma\ \textit{NCIHA}\ \Delta) \wedge (\Delta\ \textit{NCIHA}\ \Gamma)$$

end of definition

The predicate Γ ***NCIH*** Δ, is the formalization of *there is absence of computation interference hazard for components* Γ *and* Δ *that have a direct connection*. From definition 3.8, "***NCIH***" follows the symmetry of ***NCIH***.

property 3.9 *symmetry of* ***NCIH***

For i/o-connectable components Γ and Δ,

$$\Gamma\ \textit{NCIH}\ \Delta = \Delta\ \textit{NCIH}\ \Gamma$$

end of property

To illustrate definition 3.8, "***NCIH***", we present some examples.

example 3.10

We consider components Γ_w, see example 2.47, and Γ_4. Γ_4 models a "Wire with Initial Transition" element, see example 2.47. They are defined by:

$$\mathbf{o}\Gamma_w = \{b\} \quad \mathbf{i}\Gamma_w = \{a\} \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_w) = \mathbf{pref}(\{ab\}^*)$$
$$\mathbf{o}\Gamma_4 \stackrel{\text{def}}{=} \{a\} \quad \mathbf{i}\Gamma_4 \stackrel{\text{def}}{=} \{b\} \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_4) \stackrel{\text{def}}{=} \mathbf{pref}(\{ab\}^*)$$

We notice that Γ_4 ***NCIHA*** Γ_w and Γ_w ***NCIHA*** Γ_4; from definition 3.8, "***NCIH***", we conclude that Γ_w ***NCIH*** Γ_4.

end of example

example 3.11

We consider components Γ_c, cf. example 2.48, and Γ_5; they are given by:

$$\mathbf{o}\Gamma_c = \{c\} \qquad \mathbf{i}\Gamma_c = \{a, b\} \qquad \mathbf{t}(\mathbf{ptr}\,\Gamma_c) = \mathbf{pref}(\{abc, bac\}^*)$$
$$\mathbf{o}\Gamma_5 \stackrel{\text{def}}{=} \{a, b\} \qquad \mathbf{i}\Gamma_5 \stackrel{\text{def}}{=} \{c\} \qquad \mathbf{t}(\mathbf{ptr}\,\Gamma_5) \stackrel{\text{def}}{=} \mathbf{pref}(\{abc\}^*)$$

We notice that Γ_5 ***NCIHA*** Γ_c. We see that $\mathbf{ptr}\,\Gamma_5 \subseteq \mathbf{ptr}\,\Gamma_c$; nevertheless, we conclude from definition 3.5, "***NCIHA***", that Γ_c ***NCIHA*** Γ_5. From definition 3.8, "***NCIH***", we conclude that Γ_c ***NCIH*** Γ_5. Notice that we infer that Γ_c ***NCIH*** Γ_5 although $\mathbf{t}(\mathbf{ptr}\,\Gamma_c) \neq \mathbf{t}(\mathbf{ptr}\,\Gamma_5)$.

end of example

From definition 3.8, "***NCIH***" and definition 2.36, "reflection of component" we infer property 3.12.

property 3.12

For component Γ,

$$\Gamma \; \textit{NCIH} \; \overline{\Gamma}$$

end of property

3.2.1 Acceptance of commsigs

In subsection 2.1.3 we introduced informally the distinction between the "acceptance" and the "reception" of a signal. In subsection 2.1.4 we modeled this in our Communication Model by distinguishing the acceptance and the reception of a commsig by a module. In subsection 2.2.3 we argued that we also distinguish the acceptance and the reception of a commsig by a component. We now define that a *component accepts a commsig* if and only if it receives this commsig without an occurrence of computation interference. As a consequence, a module accepts a commsig if both the module receives this commsig and the occurrence of the comminst that represents the reception of this commsig is in accordance with the communication behavior of the module. A component directly controls the production of a commsig and the sending of it; however, a component has no direct control over the production of the commsigs that it receives. The sending of a commsig is independent of whether this commsig can be accepted by another component. A component has to cooperate in order to send or accept commsigs; it 'undergoes' the reception of commsigs.

remark 3.13

We say that a component *engages* in a comminst, which represents the sending or the reception of a commsig, if it either sends or accepts this commsig.

end of remark

In the remainder of this monograph we abbreviate "a component engages in a comminst that represents the sending of a commsig" into "a component sends a comminst". Analogously, we abbreviate "a component receives a commsig, the act of reception of which is represented by a comminst" into "a component receives a comminst". We also abbreviate "a component engages in a comminst that represents the acceptance of a commsig" into "a component accepts a comminst".

3.3 Transformation into computation interference hazard

We have seen in remark 3.4 that computation interference hazard amounts to "a component is not able to accept a commsig that it may receive at one of its input commports". In this section, we present a technique called *transformation into computation interference hazard*. By this technique, we 'transform' "undesired phenomenon hazards" into computation interference hazard: we establish that we deal with the undesired phenomenon hazard whenever we deal with computation interference hazard. In this way we reduce the number of undesired phenomenon hazards with which we have to deal. In subsection 3.3.0 we present the technique; in subsection 3.3.1 we give an example how this technique is applied.

3.3.0 The technique

The technique "transformation into computation interference hazard" consists of two steps:

(1) Find a trace structure T, an alphabet A, and a trace set S such that
   (i) T is the trace structure of a component,
   (ii) A is the input alphabet of this component, and
   (iii) S is the trace set that is associated with the (undesired) phenomenon hazard.

(2) Calculate **redts**(T, A, S).

We consider a component, say Γ. By replacing $\mathbf{ptr}\,\Gamma$ by $\mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\Gamma, S)$, see definition 1.34, we achieve absence of the phenomenon hazard with which S is associated, whenever absence of computation interference hazard is established. In this technique we choose for A the input alphabet of the component, since the environment of this component directly controls the 'production' of the commsigs that it sends to the component, but it has no direct control over the 'production' of the commsigs that the component sends to it.

There is an initial problem when applying this transformation technique. In order to interpret $\mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\Gamma, S)$ as the trace structure of a component, we have to guarantee that $\mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\Gamma, S)$ is non-empty, cf. property 1.40. In other words: the empty trace ε must not be removed when computing $\mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\Gamma, S)$.

theorem 3.14

Let UndesPh be some undesired phenomenon. Let trace set S be associated with UndesPh. Let Γ be a component such that $(\mathbf{A}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \cap S : I(s \restriction \mathbf{i}\Gamma) > 0)$. We define component Γ' by $\Gamma' \stackrel{\text{def}}{=} <\mathbf{io}\,\Gamma, \mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\Gamma, S)>$.

Then Γ' is the maximal (w.r.t. trace structure inclusion) component such that

(i) $\mathbf{io}\,\Gamma' = \mathbf{io}\,\Gamma$,

(ii) $\mathbf{ptr}\,\Gamma' \subseteq \mathbf{ptr}\,\Gamma$,

(iii) Γ' has absence of UndesPh hazard.

end of theorem

In theorem 3.14 component Γ' has absence of UndesPh hazard, since no traces that are associated with UndesPh are in $\mathbf{t}(\mathbf{ptr}\,\Gamma')$. For every component Δ, such that $\Delta\, \mathit{NCIH}\, \Gamma'$, we conclude using definition 1.34, that $\Delta\, \mathit{NCIH}\, \Gamma$; furthermore, using theorem 3.14 we find that Γ has absence of UndesPh hazard when communicating with such a Δ.

In subsection 3.3.1 we present an example of this transformation technique. In chapter 5 we shall transform transmission interference hazard in the communication between a component and its environment into computation interference hazard. In chapter 6 we shall transform transmission and computation interference hazard in the (internal) communication between two components into computation interference hazard at the (external inputs of the) composition of these components.

3.3.1 Example of transformation technique

In this subsection we show how we apply the technique "transformation into computation interference hazard". We choose to transform "ambiguous quiescence hazard" into computation interference hazard. In subsection 3.3.1.0 we explain the notion "ambiguous quiescence hazard" and we argue why one may be interested in it. In subsection 3.3.1.1 we show how we transform ambiguous quiescence hazard into computation interference hazard. Examples of ambiguous quiescence hazard and its transformation into computation interference hazard are shown in subsection 3.3.1.2. The notion "ambiguous quiescence hazard" was introduced in [Schols88] under the name "unspecified termination hazard". The correctness concern "absence of ambiguous quiescence hazard" is a *liveness* property.

3.3.1.0 Ambiguous quiescence hazard

We have noticed in remark 2.16 that a mechanism has no obligation to send output signals. As a consequence, a component has no obligation to send commsigs. This turns a *Molnar's-universal-do-nothing-wrong-component* (with the appropriate iobip) into an acceptable (i.e. free of computation interference hazard when connected to any i/o-connectable environment) implementation of any specification, see example 3.15. The mechanism that is modeled by such a component accepts every input signal that it may receive. Unfortunately, in general it isn't very useful, since it doesn't produce any output signal.

example 3.15 *Molnar's-universal-do-nothing-wrong-component*

Given iodir Φ, we consider component $\Gamma_{DNW(\Phi)}$; its iobip and its trace structure are defined by:

$$\mathbf{io}\,\Gamma_{DNW(\Phi)} \stackrel{\text{def}}{=} \mathbf{io}\,\Phi$$

$$\mathbf{ptr}\,\Gamma_{DNW(\Phi)} \stackrel{\text{def}}{=} < \mathbf{a}\Phi\,,\ \mathbf{pref}((\mathbf{i}\Phi)^*) >$$

This component can do nothing wrong: it accepts every commsig that it may receive; however, it doesn't send any commsig.

end of example

The correctness concern "absence of *ambiguous quiescence hazard*" amounts to "a component that is allowed to engage in an output comminst will eventually engage in some output comminst, unless it engages in an input comminst". In other words: a component has absence of ambiguous quiescence hazard if it will not stop engaging in comminsts in a state in which it is allowed to engage in an output comminst.

remark 3.16

By discussing ambiguous quiescence hazard we step outside the scope of our Communication Model: for each trace in the trace set of a component we indicate whether it is guaranteed 'to be extended' or not. One point at which ambiguous quiescence hazard might be introduced is the abstraction from module to component in subsection 2.2.3. We demonstrate this in example 3.17.

end of remark

example 3.17

We consider module Δ_6. It has one output commport γ and two input commports α and β. At most one comminst of each commport can occur. When α_1 and β_1 occur independently, no output comminst occurs. When either α_1 occurs before β_1 or β_1 occurs before α_1, output comminst γ_1 occurs thereafter:

$$\Psi^o_{\Delta_6} = \{\gamma\}$$
$$\Psi^i_{\Delta_6} = \{\alpha, \beta\}$$
$$\begin{aligned}\mathbf{CB}\,\Delta_6 = \{&<\varnothing,\varnothing>, <\{\alpha_1\},\varnothing>, <\{\beta_1\},\varnothing>, <\{\alpha_1,\beta_1\},\varnothing>\\ &, <\{\alpha_1,\beta_1\},\{\alpha_1 \sqsubset \beta_1\}>, <\{\alpha_1,\beta_1\},\{\beta_1 \sqsubset \alpha_1\}>\\ &, <\{\alpha_1,\beta_1,\gamma_1\},\{\alpha_1 \sqsubset \beta_1, \beta_1 \sqsubset \gamma_1\}>, <\{\alpha_1,\beta_1,\gamma_1\},\{\beta_1 \sqsubset \alpha_1, \alpha_1 \sqsubset \gamma_1\}>\\ &\}\end{aligned}$$

Module Δ_6 has absence of ambiguous quiescence hazard. We associate symbols *a*, *b*, and *c*, with commports α, β, and γ, respectively. Let component Γ_6 be the equivalence class of which module Δ_6 is a member. The state graph of Γ_6 is shown in figure 3.2.

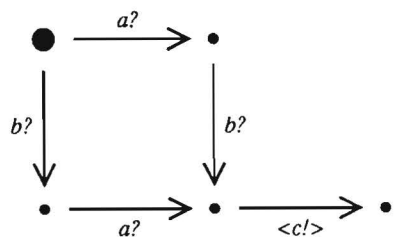

figure 3.2
State graph of component Γ_6.

In figure 3.2 we have put the label *c!* between angle brackets to indicate that we cannot guarantee that comminst γ_1 takes place. Thus, component Γ might engage in output comminst γ_1, but we cannot guarantee that it engages in any comminst. We conclude that component Γ_6 has ambiguous quiescence hazard.

One might argue that the introduction of ambiguous quiescence hazard shown in this example has more to do with the modeling in our Communication Model than with the abstraction from module Δ_6 to component Γ_6. We discuss this in section 7.1.

end of example

Ambiguous quiescence hazard, see also unspecified termination hazard in [Schols88], is related to "livelock" and "deadlock", cf. [Kimura79, Kaldewaij86]; it is an example of a liveness property that can be expressed in finite trace theory. In order to deal with more sophisticated liveness properties, finite trace theory has been extended, e.g. with refusal sets, cf. [Hoare85, Verhoeff86], or with infinite traces, cf. [Van Horn86, Black86].

3.3.1.1 Transformation of ambiguous quiescence hazard

We consider component Γ with trace structure $\mathbf{ptr}\,\Gamma$ and (input) alphabet $\mathbf{i}\Gamma$. We want to transform Γ into a component that has absence of ambiguous quiescence hazard. We need to calculate **redts**, see subsection 3.3.0. In order to apply **redts** we need a trace set, say S, that is associated with ambiguous quiescence hazard in Γ. S is defined as the set of all traces t such that

(i) t can be extended with a symbol of $\mathbf{o}\Gamma$
(formally: $(\mathbf{E}\, b : b \in \mathbf{o}\Gamma : tb \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$),

(ii) we cannot guarantee that t will be extended (with a symbol of $\mathbf{a}\Gamma$).

Let Γ' be the component such that $\mathbf{io}\Gamma' = \mathbf{io}\Gamma$ and $\mathbf{ptr}\,\Gamma' = \mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\Gamma, S)$. Component Γ' has absence of ambiguous quiescence hazard. If we connect component Γ only to components Δ for which $\Delta\, \mathit{NCIH}\, \Gamma'$, then Γ will not enter a state in which it can stop engaging in comminsts although it is allowed to engage in an output comminst.

remark 3.18

We consider components Γ and Γ' and trace set S such that $\mathbf{io}\Gamma' = \mathbf{io}\Gamma$, $\mathbf{ptr}\Gamma' = \mathbf{redts}(\mathbf{ptr}\Gamma, \mathbf{i}\Gamma, S)$, and S is associated with ambiguous quiescence hazard in Γ. If we connect Γ only to components Δ (environment of Γ) for which $\Delta \mathit{NCIH} \Gamma'$, then no instance of ambiguous quiescence will occur in component Γ. In this case component Γ behaves like component Γ', since it can only engage in comminsts in which also Γ' can engage. Component Γ' has absence of ambiguous quiescence hazard.

end of remark

In remark 3.18 we see that after reducing Γ (to Γ') we establish absence of ambiguous quiescence hazard for Γ by establishing absence of computation interference hazard (by $\Delta \mathit{NCIH} \Gamma'$).

3.3.1.2 Examples

In this subsection we present some examples of ambiguous quiescence hazard. In the diagrams of the state graphs of components we will indicate which output transitions cannot be guaranteed to take place (not even if no other transitions take place) by putting their labels between angle brackets. A state is called *lazy* if and only if

(i) it has at least one outgoing output transition, and

(ii) all its outgoing output transitions have labels between angle brackets.

In the diagrams of the state graphs of components we will mark lazy states by "*L*". We will use these extensions of state graphs only in this subsection (3.3.1.2).

We present a small example of ambiguous quiescence hazard and its transformation into computation interference hazard in example 3.19.

example 3.19

Component Γ_7 has one input commport, to which a is associated, and one output commport, to which b is associated. Initially, Γ_7 accepts an input comminst; thereafter it will produce an output comminst, unless it first receives a second input comminst: if Γ_7 receives a second input comminst before it has produced an output comminst, it will either produce two output comminsts or it will not produce any output comminst at all. If Γ_7 receives a second input comminst after it has produced an output comminst, it will produce a second output comminst thereafter. We notice that Γ_7 accepts the second input comminst anyway, but its reaction to it depends on whether it received this input comminst before or after it has sent an output comminst. The state graph of Γ_7 is shown in figure 3.3.

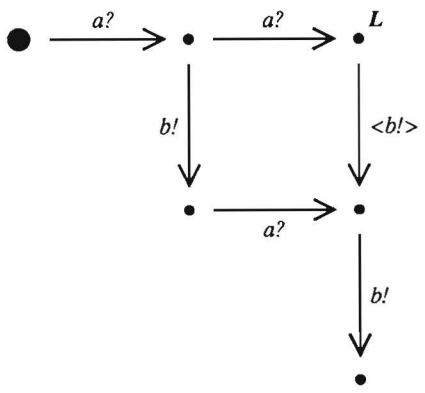

figure 3.3
State graph of component Γ_7.

In figure 3.3 the arc leaving state [*aa*] (see subsection 1.4.2) has been labeled with "<*b!*>". This means that we cannot guarantee that the transition b will take place. Since b is an element of $\mathbf{o}\Gamma_7$, state [*aa*] is lazy. For this reason it has been labeled with "***L***". State [*aa*] is the only lazy state of Γ_7. Trace *aa* is the only trace leading from the initial state to state [*aa*]. From this follows that set {*aa*} is the trace set that is associated with ambiguous quiescence hazard in Γ_7. We now calculate $\mathbf{redts}(\mathbf{ptr}\,\Gamma_7, \mathbf{i}\Gamma_7, \{aa\})$. We consider component Γ_7', that is defined by:

$$\mathbf{o}\Gamma_7' \stackrel{\text{def}}{=} \{b\}, \qquad \mathbf{i}\Gamma_7' \stackrel{\text{def}}{=} \{a\},$$

$$\mathbf{t}(\mathbf{ptr}\,\Gamma_7') \stackrel{\text{def}}{=} \mathbf{redts}(\mathbf{ptr}\,\Gamma_7, \mathbf{i}\Gamma_7, \{aa\}).$$

The state graph of Γ_7' is shown in figure 3.4.

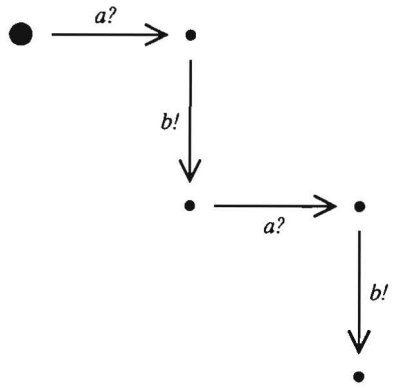

figure 3.4
State graph of component Γ_7'.

For any component, say Δ_7, such that Δ_7 ***NCIH*** Γ_7', we notice that Δ_7 ***NCIH*** Γ_7 and Γ_7 has absence of UndesPh hazard when communicating with Δ_7. We see that the transformation of Γ_7 into Γ_7' has transformed ambiguous quiescence hazard into computation interference hazard: if absence of computation interference is guaranteed between any component Δ_7 and Γ_7', Γ_7 has no ambiguous quiescence hazard, when Γ_7 communicates with such a Δ_7.

end of example

In example 3.20 we show that not only traces of the trace set that is associated with the undesired phenomenon hazard are removed, but that also prefixes thereof may be removed.

example 3.20

Component Γ_8 has one input commport, to which a is associated, and one output commport, to which b is associated. Component Γ_8 is given by figure 3.5.

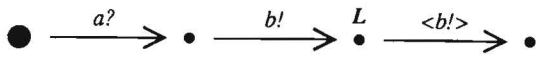

figure 3.5
State graph of component Γ_8.

After Γ_8 has received an input, it produces an output; however, thereafter it may or may not produce a second output. We notice that trace ab can be extended with b, which is an element of $\mathbf{o}\Gamma_8$, but that we cannot guarantee that ab will be extended by a symbol of $\mathbf{o}\Gamma_8$. Since ab is the only trace for which this is the case, trace set $\{ab\}$ is the trace set that is associated with ambiguous quiescence hazard in Γ_8. We notice that there is no problem with trace ab itself, but the problem has to do with extending ab. We consider component Γ_8'; it is defined by:

$$\mathbf{o}\Gamma_8' \stackrel{\text{def}}{=} \{b\}, \qquad \mathbf{i}\Gamma_8' \stackrel{\text{def}}{=} \{a\},$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma_8') \stackrel{\text{def}}{=} \mathbf{redts}(\mathbf{ptr}\,\Gamma_8, \mathbf{i}\Gamma_8, \{ab\}).$$

The state graph of the trace set of Γ_8' is shown in figure 3.6.

figure 3.6
State graph of $\mathbf{t}(\mathbf{ptr}(\Gamma_8'))$.

We see that by transforming Γ_8 into Γ_8' not only trace ab is removed from the trace set, but also trace a.

end of example

The following examples are more realistic and more complex.

Example 3.21 is spread over two pages. It starts at page 94.

example 3.21

We consider component Γ_9 that is given by figure 3.7.

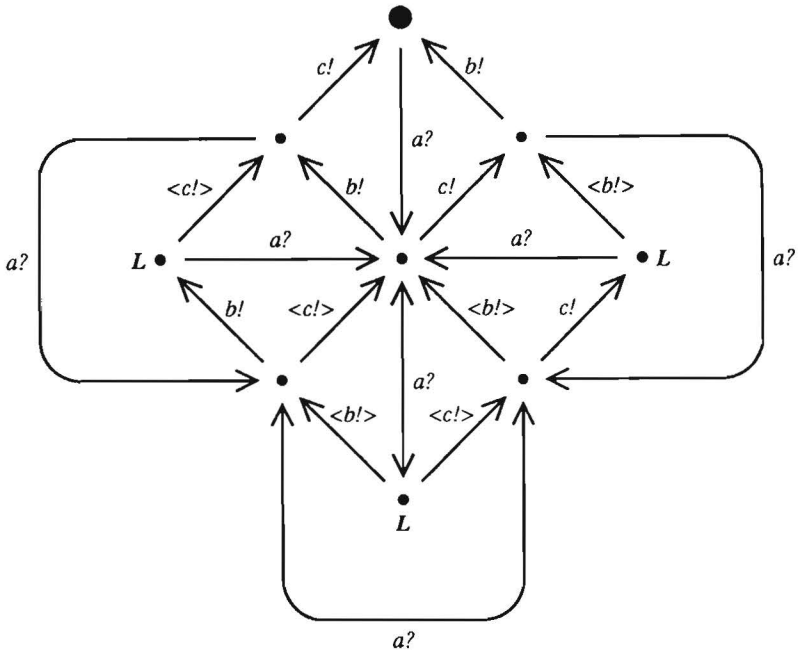

figure 3.7
State graph of component Γ_9.

When Γ_9 is in a state labeled with "L" it may or may not produce an output; in all other states in which it is allowed to produce an output, it will eventually produce it. Let S_9 be the set of all traces that lead to a state labeled with "L" in figure 3.7. Now, S_9 is the trace set that is associated with ambiguous quiescence hazard in Γ_9. We consider component Γ_9', that is defined by:

$$\mathbf{o}\Gamma_9' \stackrel{\text{def}}{=} \mathbf{o}\Gamma_9, \qquad \mathbf{i}\Gamma_9' \stackrel{\text{def}}{=} \mathbf{i}\Gamma_9,$$

$$\mathbf{t}(\mathbf{ptr}\,\Gamma_9') \stackrel{\text{def}}{=} \mathbf{redts}(\mathbf{ptr}\,\Gamma_9, \mathbf{i}\Gamma_9, S_9).$$

The state graph of Γ_9' is shown in figure 3.8. We notice that $\Gamma_9' = \Gamma_f$, cf. example 2.49.

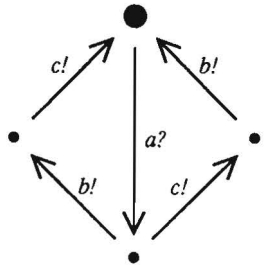

figure 3.8
State graph of component Γ_9'.

Again, the transformation of Γ_9 into Γ_9' has transformed ambiguous quiescence hazard into computation interference hazard.

end of example

example 3.22

We consider component Γ_{10} that is given by figure 3.9. Component Γ_{10} models some kind of "Or" element (see example 2.51): $\mathbf{ptr}\,\Gamma_{or} \subseteq \mathbf{ptr}\,\Gamma_{10}$. The difference between them is that Γ_{10} may at some points engage in two output comminsts of the same output commport whereas in such a case Γ_{or} doesn't engage in any output comminst at all.

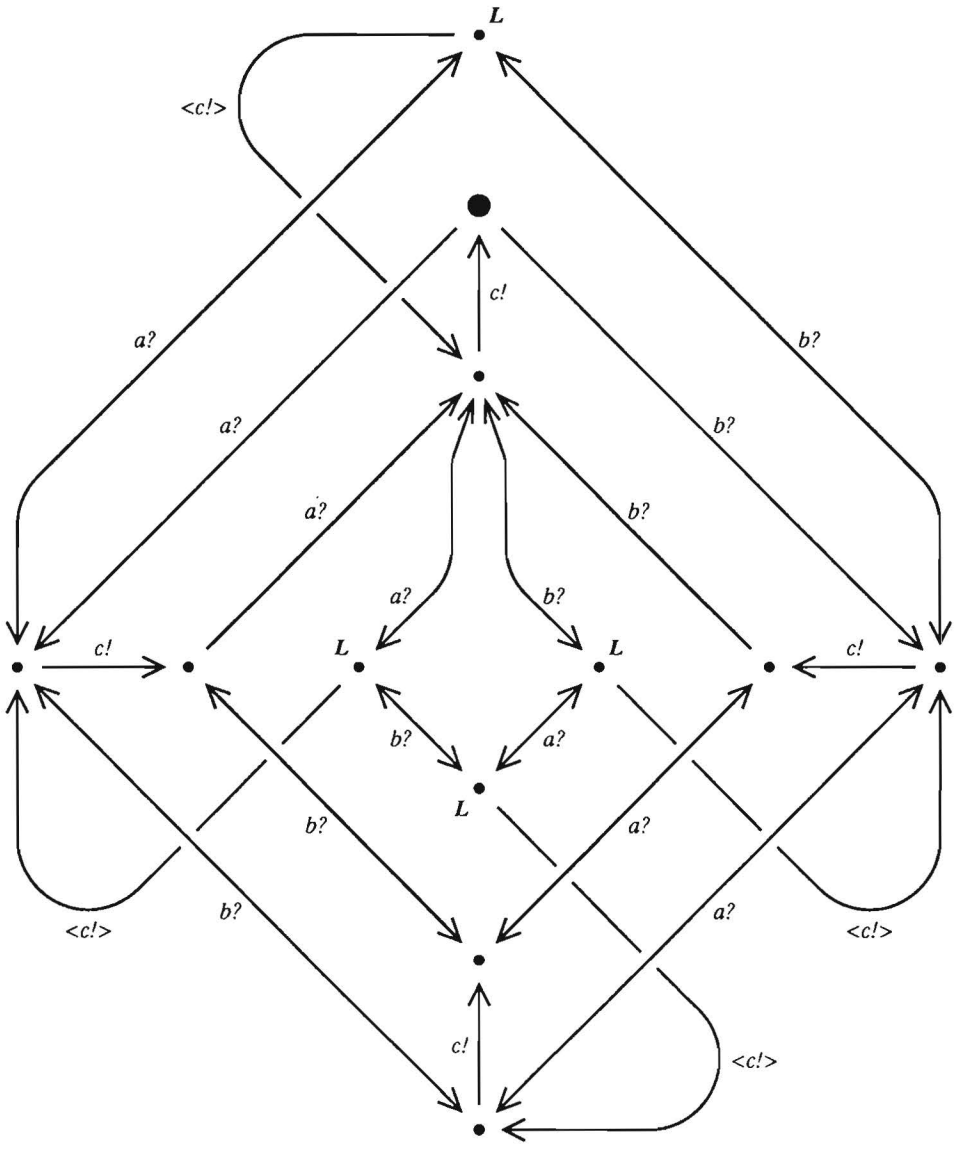

figure 3.9
State graph of component Γ_{10}.

Let S_{10} be the trace set that is associated with ambiguous quiescence hazard in Γ_{10}, viz. the set of all traces that lead from the initial state to one of the four states labeled with "***L***" in figure 3.9. We consider component Γ'_{10}, that is defined by:

$$\mathbf{o}\Gamma'_{10} \stackrel{\text{def}}{=} \mathbf{o}\Gamma_{10}, \qquad \mathbf{i}\Gamma'_{10} \stackrel{\text{def}}{=} \mathbf{i}\Gamma_{10},$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma'_{10}) \stackrel{\text{def}}{=} \mathbf{redts}(\mathbf{ptr}\,\Gamma_{10}, \mathbf{i}\Gamma_{10}, S_{10}).$$

The state graph of Γ'_{10} is shown in figure 3.10.

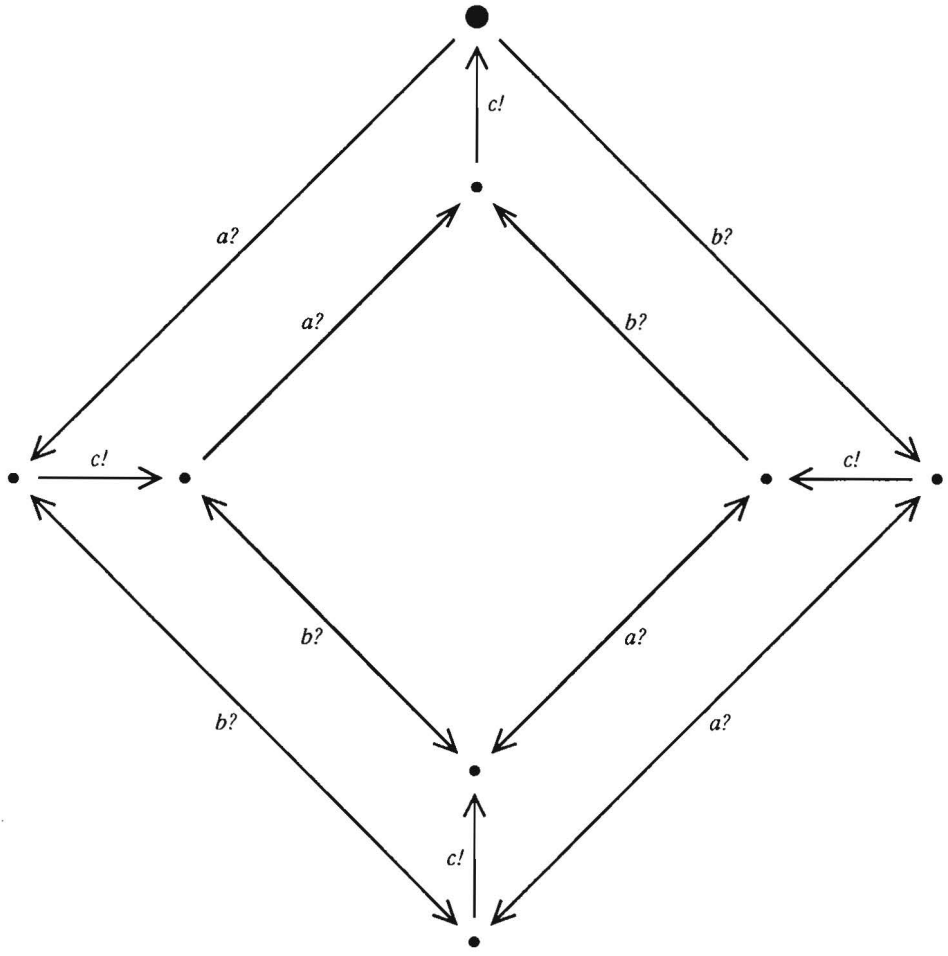

figure 3.10
State graph of component Γ'_{10}.

Again, the transformation of Γ_{10} into Γ'_{10} has transformed ambiguous quiescence hazard into computation interference hazard.

end of example

4

Communicating delay-safely

In this chapter we address *indirect connections*. In an indirect connection matching commports model terminals that are connected by a wire, see subsection 2.1.0.

Our formal definition of the delay-safety of a channel is based on our *causality* notion:

> no commsig is received before it has been sent.

This causality notion models that there is only one assumption made with respect to the delay of a signal that is sent from one terminal via a wire to another terminal in the physical model, viz.:

> the value of this delay is nonnegative.

Even distinct signals that travel along the same wire may have different values of delays.

Notice that delay-safety is not a property of a component, but it is a property of a channel. 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.

In this chapter we introduce three important operators in our Communication Model. In subsection 4.1.1 we present the delay-safe closure of a channel. Given channel Θ, channel $\mathbf{DSC}\,\Theta$ is the smallest (w.r.t. trace structure inclusion) delay-safe channel such that $\mathbf{ptr}\,\Theta \subseteq \mathbf{ptr}(\mathbf{DSC}\,\Theta)$. In subsection 4.2.1 we present **DSE**,

i.e. the delay-safe enclosure of a component. For component Γ, component $\overline{\mathbf{DSE}\,\Gamma}$ is the maximal (w.r.t. trace structure inclusion) partner of Γ; when Γ and $\overline{\mathbf{DSE}\,\Gamma}$ are indirectly connected, they have no computation interference hazard. In subsection 4.2.3 we present **CBDS**, i.e. the communication behavior of a delay-safely communicating component. The maximal (w.r.t. trace structure inclusion) communication behavior of a component, say Γ, that communicates delay-safely without computation interference hazard equals trace structure $\mathbf{cbds}\,\Gamma$ ($\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Gamma$). This means that Γ behaves in that case like component $\mathbf{CBDS}\,\Gamma$ ($\mathbf{io}(\mathbf{CBDS}\,\Gamma) = \mathbf{io}\,\Gamma$ and $\mathbf{ptr}(\mathbf{CBDS}\,\Gamma) = \mathbf{cbds}\,\Gamma$).

4.0 Causality

In this section we formalize our causality notion. We consider the components Γ and Δ such that $\mathbf{io}\,\overline{\Gamma} = \mathbf{io}\,\Delta$; as a consequence, Γ and Δ have a closed connection, i.e. $\mathbf{a}\Gamma = \mathbf{a}\Delta$. Let t and u be traces such that $t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $u \in \mathbf{t}(\mathbf{ptr}\,\Delta)$. In chapter 3 we have considered components that have a direct connection; in that case, if t and u are consistent, they are equal. In this chapter we deal with components that are indirectly connected; now, t and u need not be equal; the condition that t and u have to satisfy is called *composability*. In figure 4.0 we show these two components that have an indirect connection.

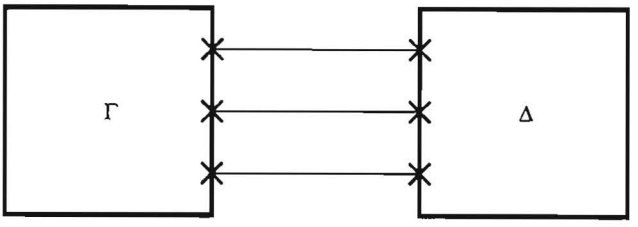

figure 4.0
Components Γ and Δ that have an indirect connection.

We have stated in the beginning of this chapter that no commsig can be received before it has been sent. In order to model this causality we define the *composability* relation between traces. Let iobip F be such that $F = \mathbf{io}\,\Gamma$. Let t and u be traces such that $t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $u \in \mathbf{t}(\mathbf{ptr}\,\Delta)$. We call t *composable under* F with u, if, at some moment, t is associated with a comminstorder of Γ and u is associated with a comminstorder of Δ such that t and u are consistent with our causality notion, cf. also subsection 4.0.2.

remark 4.0

Molnar has characterized "composability" in a nice way:

> Trace t is a member of a trace set that is associated with a comminstorder of Γ. Trace u is a member of a trace set that is associated with a comminstorder of Δ. Causality implies a partial order between comminsts of Γ and Δ. Composability of t and u equals the existence of a full order consistent with the union of these three partial orders (viz., the two comminstorders and the partial order that is implied by causality).

end of remark

Initially, when no comminsts have happened yet, both t and u are equal to ε. From a pair of composable traces we construct another pair of composable traces by extending one of the traces with one symbol. Since no commsig can be received before it has been sent, the extension of a trace with an input symbol is restricted, see definition 4.1.

definition 4.1 *composability*

Given are traces t and u, and iobip F such that $t \in (\mathbf{a}F)^*$ and $u \in (\mathbf{a}F)^*$; we define that t *is composable under* F with u, denoted by $t\mathbf{c}_F u$, recursively by

(i) $\varepsilon \mathbf{c}_F \varepsilon$

(ii) for traces t and u and symbol a such that $t\mathbf{c}_F u$ and $a \in \mathbf{o}F$,

$$ta\mathbf{c}_F u$$

(iii) for traces t and u and symbol a such that $t\mathbf{c}_F u$, $a \in \mathbf{o}F$, and $\#_a t > \#_a u$,

$$t\mathbf{c}_F ua$$

(iv) for traces t and u and symbol b such that $t\mathbf{c}_F u$ and $b \in \mathbf{i}F$,

$$t\mathbf{c}_F ub$$

(v) for traces t and u and symbol b such that $t\mathbf{c}_F u$, $b \in \mathbf{i}F$, and $\#_b u > \#_b t$,

$$tb\mathbf{c}_F u$$

(vi) completeness axiom: t is not composable under F with u, unless this is required by (i), (ii), (iii), (iv), or (v).

end of definition

The conditions in the definition above reflect that no commsig is received before it has been sent, see also subsection 4.0.2.

Udding was the first to define composability formally, cf. [Udding84]. Composability as defined in definition 4.1 is equal to composability as defined by Verhoeff, cf. [Verhoeff85]. In definition 4.1 no trace set or trace structure is involved: only the iobip is important. The earlier definitions in [Udding84] and [Schols85] restrict the traces to elements of given trace sets. When this restriction is dropped, all definitions are equivalent, see [Schols85] and [Verhoeff85]. The present definition is nicer from a mathematical point of view, cf. [Verhoeff85], than the definitions in [Udding84] and [Schols85]. Property 4.2 asserts that the non-restricted version of the definition in [Schols85] is equivalent to definition 4.1; for a proof of this property we refer to [Siccama86].

property 4.2 *composability*

For traces t and u, and iobip F such that $t \in (\mathbf{a}F)^*$ and $u \in (\mathbf{a}F)^*$,

$$
\begin{aligned}
t \mathbf{c}_F u = (\ & (\mathbf{A}\, a : a \in \mathbf{o}F : \#_a t \geq \#_a u) \\
& \wedge (\mathbf{A}\, b : b \in \mathbf{i}F : \#_b u \geq \#_b t) \\
& \wedge (\mathbf{A}\, a, b, r, s : a \in \mathbf{o}F \wedge b \in \mathbf{i}F \wedge r b \mathbf{prefix} t \wedge s a \mathbf{prefix} u \\
& \qquad : (\#_a r > \#_a s) \vee (\#_b s > \#_b r) \\
&)\ \\
&)
\end{aligned}
$$

end of property

Unfortunately, none of the definitions of composability mentioned above is very well suited to check manually whether two traces are composable under an iobip. For this reason, we present Verhoeff's method to check this graphically, see subsection 4.0.0.

4.0.0 Composability diagram

Whether a trace t is composable under an iobip F with a trace u, can be concluded by constructing a *composability diagram*. Such a diagram provides more insight into the composability relation, and its construction is a practical tool for concluding whether t and u are composable under F or not.

The symbols in trace t are listed in the top row; each symbol is postfixed with an exclamation mark or a question mark to indicate whether it is an element of $\mathbf{o}F$ or $\mathbf{i}F$, respectively. The symbols in trace u are listed in the bottom row; each symbol is postfixed with an exclamation mark or a question mark to indicate whether it is an element of $\mathbf{o}\bar{F}$ or $\mathbf{i}\bar{F}$, respectively. To the right of the last (right most) symbol of each trace an end of trace marker (\$) is added.

For every symbol, its first (left most) occurrence in t is connected to its first occurrence in u by an arrow pointing from the occurrence that is postfixed with an exclamation mark to the occurrence that is postfixed with a question mark (if there are not enough occurrences in either one of the traces, the \$ at the end of that trace is used instead). The second and higher occurrences of symbols in t or u are connected in the same way. Now, all occurrences of symbols are connected by some arrow. See figures 4.1 and 4.3 for such a composability diagram.

In a composability diagram two intersecting arrows are said to form a *backward intersection* if and only if one arrow (the *tu*-arrow) starts at trace t and the other arrow (the *ut*-arrow) starts at trace u, the *tu*-arrow points in trace u to the left of the beginning of the *ut*-arrow, and the *ut*-arrow points in trace t to the left of the beginning of the *tu*-arrow.

Trace t is composable under iobip F with trace u if and only if in the composability diagram:

(i) there is no arrow starting from a \$, and

(ii) there is no backward intersection of two arrows.

example 4.3 *composable traces*

We consider traces t and u, symbols a, b, c, and d, and iobip F_0 such that $\mathbf{o}F_0 = \{a, c\}$ and $\mathbf{i}F_0 = \{b, d\}$. We are interested in whether trace $abca$ $(=t)$ is composable under F_0 with trace $adbcb$ $(=u)$. In figure 4.1 this composability diagram is shown.

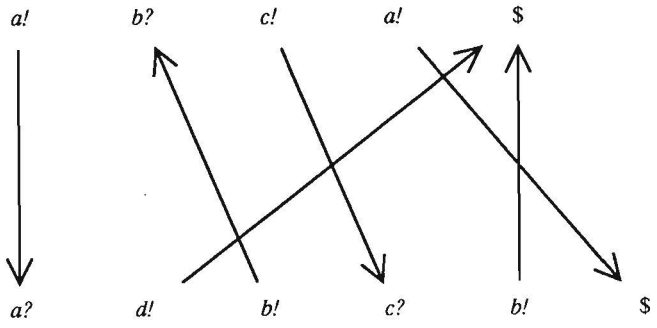

figure 4.1
Composability diagram.

The absence of both an arrow starting from a \$ and a backward intersection of two arrows in the composability diagram indicates that t and u are composable under F_0. By direct application of definition 4.1, "composability", we can derive in several ways a confirmation that $abca\mathbf{c}_{F_0}adbcb$:

$\varepsilon\mathbf{c}_{F_0}\varepsilon$	$\varepsilon\mathbf{c}_{F_0}\varepsilon$	$\varepsilon\mathbf{c}_{F_0}\varepsilon$
$a\mathbf{c}_{F_0}\varepsilon$	$a\mathbf{c}_{F_0}\varepsilon$	$a\mathbf{c}_{F_0}\varepsilon$
$a\mathbf{c}_{F_0}a$	$a\mathbf{c}_{F_0}a$	$a\mathbf{c}_{F_0}a$
$a\mathbf{c}_{F_0}ad$	$a\mathbf{c}_{F_0}ad$	$a\mathbf{c}_{F_0}ad$
$a\mathbf{c}_{F_0}adb$	$a\mathbf{c}_{F_0}adb$	$a\mathbf{c}_{F_0}adb$
$ab\mathbf{c}_{F_0}adb$	$ab\mathbf{c}_{F_0}adb$	$ab\mathbf{c}_{F_0}adb$
$abc\mathbf{c}_{F_0}adb$	$abc\mathbf{c}_{F_0}adb$	$abc\mathbf{c}_{F_0}adb$
$abc\mathbf{c}_{F_0}adbc$	$abc\mathbf{c}_{F_0}adbc$	$abca\mathbf{c}_{F_0}adb$
$abc\mathbf{c}_{F_0}adbcb$	$abca\mathbf{c}_{F_0}adbc$	$abca\mathbf{c}_{F_0}adbc$
$abca\mathbf{c}_{F_0}adbcb$	$abca\mathbf{c}_{F_0}adbcb$	$abca\mathbf{c}_{F_0}adbcb$

table 4.2
Three derivations of $abca\mathbf{c}_{F_0}adbcb$.

end of example

We consider the case that two traces, say t and u, are not composable under an iobip, say F. Now, there must exist an arrow starting from a \$ or a backward intersection of two arrows in the composability diagram. All arrows that form some backward intersection indicate together the longest prefixes (of traces t and u) that are composable under iobip F, cf. example 4.4

example 4.4 *traces that are not composable*

We consider traces t and u, symbols a, b, c, and d, and iobip F_0 such that $\mathbf{o}F_0 = \{a, c\}$ and $\mathbf{i}F_0 = \{b, d\}$. We are interested in whether trace bca $(=t)$ is composable under F_0 with trace $acdb$ $(=u)$.

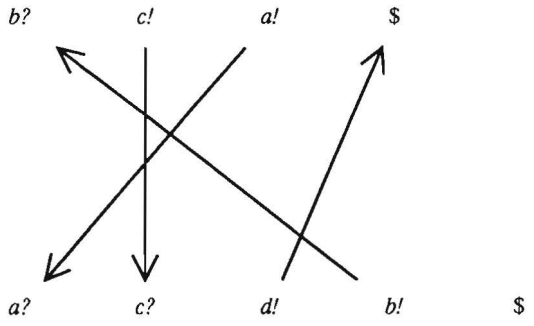

figure 4.3
Composability diagram.

In figure 4.3 this composability diagram is shown. From definition 4.1, "composability", we infer that $\neg(b\mathbf{c}_{F_0}\varepsilon)$ and $\neg(\varepsilon\mathbf{c}_{F_0}a)$. We conclude that $\neg(bca\mathbf{c}_{F_0}acdb)$. Notice that the leftmost arrowheads of all arrowheads of the arrows that form some backward intersection indicate the longest prefixes that are composable under F_0: $\varepsilon\mathbf{c}_{F_0}\varepsilon$.

end of example

This construction of the composability diagram has first been shown by Verhoeff, cf. [Verhoeff89]. He has characterized $t\mathbf{c}_F u$ by requirements (i) and (ii) (see p.103) w.r.t. the composability diagram of $t\mathbf{c}_F u$; this characterization corresponds to the (non-recursive) definition of composability in property 4.2, "composability". In this property, the first two conjuncts at the right hand side of the equation formalize requirement (i) in the construction of the composability diagram above; they reflect that each received commsig is sent at some moment before or after it is received. The third conjunct formalizes requirement (ii) in that construction; it reflects that, when two commsigs travel in opposite directions, at least one of them must have been sent when either of them is received. The first two conjuncts are insufficient to model the nonnegativeness of the delays: together with the third conjunct, they impose this requirement.

4.0.1 Properties of composability

In this subsection we present some properties of composability. These properties were given in [Udding84].

property 4.5

For traces t and u, symbols a and b, and iobip F such that $t \in (\mathbf{a}F)^*$, $u \in (\mathbf{a}F)^*$, $a \in \mathbf{a}F$, and $b \in \mathbf{a}F$,

(i) $t \mathbf{c}_F t$

(ii) $ta \mathbf{c}_F \varepsilon \Rightarrow t \mathbf{c}_F \varepsilon$

(iii) $\varepsilon \mathbf{c}_F ub \Rightarrow \varepsilon \mathbf{c}_F u$

(iv) $ta \mathbf{c}_F ub \Rightarrow (ta \mathbf{c}_F u \vee t \mathbf{c}_F ub)$

end of property

property 4.6

For trace t, symbol a, and iobip F such that $t \in (\mathbf{a}F)^*$,

(i) for $a \in \mathbf{o}F$, $\quad ta \mathbf{c}_F t$

(ii) for $a \in \mathbf{i}F$, $\quad t \mathbf{c}_F ta$

end of property

property 4.7

For traces t and u, symbol a, and iobip F such that $t \in (\mathbf{a}F)^*$, $u \in (\mathbf{a}F)^*$, and $a \in \mathbf{a}F$,

$$t \mathbf{c}_F ua \Rightarrow (\mathbf{E}\, s : s \mathbf{prefix} t : s \mathbf{c}_F u)$$

end of property

example 4.8

For symbols a and b, and iobip F_l such that $\mathbf{o}F_l = \{a\}$ and $\mathbf{i}F_l = \{b\}$,

$$ab \mathbf{c}_{F_l} ba \wedge \neg (ab \mathbf{c}_{\bar{F}_l} ba)$$

end of example

From the example above we observe that, in general, the definition of composable traces is not symmetric in inputs and outputs. (A contrary statement by Udding, see [Udding84, p.44], is erroneous; this does not invalidate Udding's work). Because of this asymmetry, an asymmetric iobip F is needed to define composability.

property 4.9

For traces t and u, and iobip F such that $t \in (\mathbf{a}F)^*$ and $u \in (\mathbf{a}F)^*$,

$$t \mathbf{c}_F u = u \mathbf{c}_{\bar{F}} t$$

end of property

Property 4.9 illustrates that pairs of composable traces do satisfy some symmetry property; as a consequence, the particular choice of the iobip with respect to the alphbip of a channel, say Θ, viz. (given symbol $a \in \mathbf{a}\Theta$) $\mathbf{i}F = \mathbf{spa}(a, \Theta)$ and $\mathbf{o}F = \mathbf{opa}(a, \Theta)$ or the other possibility $\mathbf{i}F = \mathbf{opa}(a, \Theta)$ and $\mathbf{o}F = \mathbf{spa}(a, \Theta)$, is irrelevant.

From property 4.2, "composability", we infer that composability is transitive.

property 4.10 *transitivity of composability*

For traces s, t, and u, and iobip F,

$$(s\mathbf{c}_F t \wedge t\mathbf{c}_F u) \Rightarrow s\mathbf{c}_F u$$

end of property

property 4.11

For traces t and u, symbol a, and iobip F such that $t \in (\mathbf{a}F)^*$ and $u \in (\mathbf{a}F)^*$,

(i) for $a \in \mathbf{o}F$, $\quad (t\mathbf{c}_F u \wedge \#_a t > \#_a u) = t\mathbf{c}_F ua$

(ii) for $a \in \mathbf{o}F$, $\quad t\mathbf{c}_F u = (ta\mathbf{c}_F u \wedge \#_a t \geq \#_a u)$

(iii) for $a \in \mathbf{i}F$, $\quad (t\mathbf{c}_F u \wedge \#_a t < \#_a u) = ta\mathbf{c}_F u$

(iv) for $a \in \mathbf{i}F$, $\quad t\mathbf{c}_F u = (t\mathbf{c}_F ua \wedge \#_a t \leq \#_a u)$

end of property

In property 4.11 the implications from right to left are the most important ones, since the implications from left to right are similar to those in definition 4.1, "composability". From property 4.11 we derive property 4.12.

property 4.12

For iobip F, traces t and u, and symbol a such that $t \in (\mathbf{a}F)^*$, $u \in (\mathbf{a}F)^*$, and $a \in \mathbf{a}F$,

$$t\mathbf{c}_F u = ta\mathbf{c}_F ua$$

end of property

4.0.2 Composability versus independence of comminsts

In section 4.0 we formalized our causality notion "no commsig is received before it has been sent". In this subsection we have a closer look at this formalization. We first study the following example.

example 4.13

We consider components Γ_I and Δ_I that have a closed and indirect connection, see figure 4.4.

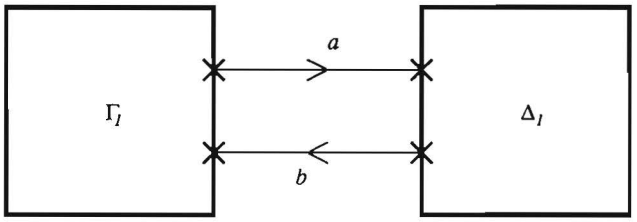

figure 4.4
Components Γ_I and Δ_I.

Component Γ_I accepts one input comminst before it sends one output comminst. Let T_I be the trace set that is associated with the comminstorder of Γ_I in which the input comminst occurs before the output comminst of Γ_I: $T_I = \{ba\}$. Component Δ_I accepts one input comminst and sends one output comminst; these comminsts occur independently. Let U_I be the trace set that is associated with the comminstorder of Δ_I in which the input comminst and output comminst of Δ_I occur independently: $U_I = \{ab, ba\}$.

$$\mathbf{o}\Gamma_I \stackrel{\text{def}}{=} \{a\}, \quad \mathbf{i}\Gamma_I \stackrel{\text{def}}{=} \{b\}, \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_I) \stackrel{\text{def}}{=} \mathbf{pref}\{ba\},$$
$$\mathbf{o}\Delta_I \stackrel{\text{def}}{=} \{b\}, \quad \mathbf{i}\Delta_I \stackrel{\text{def}}{=} \{a\}, \quad \mathbf{t}(\mathbf{ptr}\,\Delta_I) \stackrel{\text{def}}{=} \mathbf{pref}\{ab, ba\}.$$

Let iobip F_I be such that $F_I = \mathbf{io}\Gamma_I$. We notice that $\neg(ba\,\mathbf{c}_{F_I}\,ab)$ in spite of the independence of the comminsts of Δ_I.

end of example

In example 4.13 we considered the comminstorder of Γ_l, with which trace set T_l is associated, and the comminstorder of Δ_l, with which trace set U_l is associated. These comminstorders are consistent with our causality notion. Trace *ba* is an element of T_l and trace *ab* is an element of U_l. Nevertheless, these traces are not composable under F_l. This seems to be a problem: the question arises whether our composability operator, see definition 4.1, can be associated with our causality notion. We consider iobip F and trace t. We are not interested whether $t\mathbf{c}_F u$ holds for a particular trace u. But, we are interested in whether $(\mathbf{E}\, u : u \in U : t\mathbf{c}_F u)$ holds for some trace set U. Furthermore, trace set U is such that it is the union of trace sets that are associated with comminstorders, cf. subsection 2.2.3. As a consequence, if two comminstorders are consistent with our causality notion, then for every trace (t) in the trace set that is associated with one of these comminstorders there exists a trace (u) in the trace set that is associated with the order comminstorder such that these two traces are composable $(t\mathbf{c}_F u)$. This is why there is no problem in associating our composability operator with our causality notion. We illustrate this by example 4.14.

example 4.14

We consider components Γ_l and Δ_l, trace sets T_l and U_l, and iobip F_l, see example 4.13. We notice that $ba \in \mathbf{t}(\mathbf{ptr}\,\Gamma_l)$, $ba \in T_l$, $ba \in \mathbf{t}(\mathbf{ptr}\,\Delta_l)$, $ba \in U_l$, and $ba\mathbf{c}_{F_l} ba$. As a consequence, $(\mathbf{E}\, u : u \in U_l : ba\mathbf{c}_{F_l} u)$. From this we find $(\mathbf{E}\, t, u : t \in T_l \wedge u \in U_l : t\mathbf{c}_{F_l} u)$. We conclude that the comminstorders with which T_l and U_l are associated are consistent with our causality notion.

end of example

remark 4.15

Since we are only interested in predicate $(\mathbf{E}\, u : u \in U : t\mathbf{c}_F u)$ for trace t, trace set U, and iobip F, we conclude that, in this way, we may associate the composability of traces in the trace theory formalism with our causality notion in our Communication Model.

end of remark

The statement in remark 4.15 has been relied on by everyone that uses Udding's composability operator, see [Udding84], to model concurrent or parallel behavior.

4.1 Communication in channels

In this section we address the communication in channels. From subsection 2.2.4 we recall that a channel is a pair: an alphbip and a trace structure.

4.1.0 Delay-safe channels

We define the class $\mathbf{D}_4$ in order to formalize that a channel is "delay-safe", see definition 4.19.

definition 4.16 $\mathbf{D}_4$

For trace structure T and alphbip D, the pair (T, D) is an element of $\mathbf{D}_4$ if and only if T is nonempty and prefix-closed and

$$T = < \mathbf{i}F \cup \mathbf{o}F \,,\, \{s\,,t\,,u : s \in \mathbf{t}T \wedge s\mathbf{c}_F t \wedge t\mathbf{c}_F u \wedge u \in \mathbf{t}T : t\} >$$

for some iobip F such that $D = \mathbf{i}F \oplus \mathbf{o}F$.

end of definition

In definition 4.16 an iobip is needed in order to address the composability of traces; this is why there is an existential quantification over iobip F.

remark 4.17

Property 4.18 shows that we need to consider only one iobip when proving that a "trace structure" – alphbip pair is not in $\mathbf{D}_4$.

end of remark

property 4.18

For trace structure T and iobip F such that $\mathbf{a}T = \mathbf{i}F \cup \mathbf{o}F$,

$$(\mathbf{t}T \neq \{s\,,t\,,u : s \in \mathbf{t}T \wedge s\mathbf{c}_F t \wedge t\mathbf{c}_F u \wedge u \in \mathbf{t}T : t\}) \Rightarrow (T, \mathbf{i}F \oplus \mathbf{o}F) \notin \mathbf{D}_4$$

end of property

In [Schols85] we used the "Foam Rubber Wrapper Postulate" (see also remark 4.35) to give the definition of what we now call "delay-safe channel". Here we present an equivalent form of this definition, using the class $\mathbf{D}_4$, cf. [Schols85, Siccama86, Verhoeff85]. In section 4.0 we have shown that the composability of traces formalizes our causality notion ("no commsig is received before it has been sent"). Using class $\mathbf{D}_4$ we now define that *a channel is delay-safe*.

definition 4.19 *delay-safe channel*

For channel Θ, we call Θ *delay-safe* if and only if

$$(\mathbf{ptr}\,\Theta, \mathbf{ab}\Theta) \in \mathbf{D}_4$$

end of definition

4.1.1 Delay-safe closure

The operator **dsc** yields the trace structure of the mathematical closure of a "trace structure" – alphbip pair within the class $\mathbf{D}_4$.

definition 4.20 **dsc**

For trace structure T and alphbip D such that $\mathbf{a}T = \mathbf{a}D$, trace structure $\mathbf{dsc}(T,D)$ is the smallest (w.r.t. trace structure inclusion) trace structure such that

$$T \subseteq \mathbf{dsc}(T,D) \wedge (\mathbf{dsc}(T,D),D) \in \mathbf{D}_4$$

end of definition

Notice that in the definition above $\mathbf{a}T = \mathbf{a}(\mathbf{dsc}(T,D))$; this follows from definition 1.18, "trace structure inclusion". In [Schols 85] we derived that such a unique minimum exists.

property 4.21

For trace structure T and alphbip D such that $(T,D) \in \mathbf{D}_4$,

$$\mathbf{dsc}(T,D) = T$$

end of property

remark 4.22

From property 4.21 we infer that, for every alphbip D, the function $\mathbf{dsc}(T,D)$ is idempotent in T, i.e.

$$\mathbf{dsc}(\mathbf{dsc}(T,D),D) = \mathbf{dsc}(T,D)$$

end of remark

property 4.23 **dsc** *is monotonic in its first argument*

For trace structures T and U, and alphbip D such that $\mathbf{a}T = \mathbf{a}D$ and $\mathbf{a}U = \mathbf{a}D$,

$$(T \subseteq U) \Rightarrow (\mathbf{dsc}(T,D) \subseteq \mathbf{dsc}(U,D))$$

end of property

We extend the definition of **dsc** to components and channels.

definition 4.24 **dsc** *of component or channel*

For component or channel Θ, trace structure $\mathbf{dsc}\,\Theta$ is defined by

$$\mathbf{dsc}\,\Theta \stackrel{\text{def}}{=} \mathbf{dsc}(\mathbf{ptr}\,\Theta, \mathbf{ab}\Theta)$$

end of definition

property 4.25

For component Γ,

$$\mathbf{dsc}\,\Gamma = \mathbf{dsc}\,\overline{\Gamma}$$

end of property

definition 4.26 *delay-safe closure of channel*

For channel Θ the channel $\mathbf{DSC}\,\Theta$ denotes the *delay-safe closure* of Θ; it is defined by

$$\mathbf{ab}(\mathbf{DSC}\,\Theta) \stackrel{\text{def}}{=\!=} \mathbf{ab}\Theta$$

and

$$\mathbf{ptr}(\mathbf{DSC}\,\Theta) \stackrel{\text{def}}{=\!=} \mathbf{dsc}\,\Theta$$

end of definition

Given a channel Θ, channel $\mathbf{DSC}\,\Theta$ is the smallest (w.r.t. trace structure inclusion) delay-safe channel such that $\mathbf{ptr}\,\Theta \subseteq \mathbf{dsc}\,\Theta$.

remark 4.27

Given is a channel Θ that is not delay-safe. Now $\mathbf{dsc}\,\Theta$ can be associated with the communication in Θ instead of $\mathbf{ptr}\,\Theta$. This is formalized in definition 4.26, "delay-safe closure of channel".

end of remark

We have no interpretation for the "delay-safe closure of a component", cf. remark 4.59 and example 4.58. For this reason we do not define it.

From property 4.21 we derive that **DSC** is idempotent, see also remark 4.22.

property 4.28 **DSC** *is idempotent*

For channel Θ,

$$\mathbf{DSC}(\mathbf{DSC}\,\Theta) = \mathbf{DSC}\,\Theta$$

end of property

4.2 Communication behavior of components

In this section we concentrate on the communication behavior of components that communicate delay-safely. We have one correctness concern: absence of computation interference hazard. We shall show that delay-safe communication may restrict the communication behavior of a component, when absence of computation interference hazard is a correctness concern.

4.2.0 Computation interference hazard

We extend definition 3.5, "***NCIHA***", and definition 3.8, "***NCIH***", to indirect connections.

definition 4.29 *NCIHADS*

Given are i/o-connectable components Γ and Δ. Let iobip F be such that $\mathbf{i}F = \mathbf{i}\Gamma \cap \mathbf{o}\Delta$ and $\mathbf{o}F = \mathbf{o}\Gamma \cap \mathbf{i}\Delta$. By Γ ***NCIHADS*** Δ, we denote that there is *no computation interference hazard at* Δ, *when* Γ *and* Δ *have an indirect connection*; Γ ***NCIHADS*** Δ is defined by

$$\Gamma\ \textbf{\textit{NCIHADS}}\ \Delta \stackrel{\text{def}}{=} (\mathbf{A}\, t, u, a : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge u \in \mathbf{t}(\mathbf{ptr}\,\Delta) \wedge a \in \mathbf{o}F \wedge (t\restriction(\mathbf{a}\Gamma \cap \mathbf{a}\Delta))\mathbf{c}_F(u\restriction(\mathbf{a}\Gamma \cap \mathbf{a}\Delta)) \wedge \#_a t > \#_a u : ua \in \mathbf{t}(\mathbf{ptr}\,\Delta))$$

end of definition

Given that Γ and Δ have an indirect connection, the condition $(t\restriction(\mathbf{a}\Gamma \cap \mathbf{a}\Delta))\mathbf{c}_F(u\restriction(\mathbf{a}\Gamma \cap \mathbf{a}\Delta))$ in definition 4.29 reflects that t and u are consistent with our causality notion. Definition 4.29 reflects that Δ accepts every commsig that it may receive.

Using definition 4.29, "***NCIHADS***", we define absence of computation interference hazard when the connection is indirect.

definition 4.30 *computation interference hazard for indirect connection*

Given are i/o-connectable components Γ and Δ. Γ *and* Δ *have no computation interference hazard, when they have an indirect connection*, which is denoted by Γ ***NCIHDS*** Δ, is defined by

$$\Gamma\ \textbf{\textit{NCIHDS}}\ \Delta \stackrel{\text{def}}{=} (\Gamma\ \textbf{\textit{NCIHADS}}\ \Delta) \wedge (\Delta\ \textbf{\textit{NCIHADS}}\ \Gamma)$$

end of definition

"Computation interference hazard at one component for indirectly connected components" and "computation interference hazard for indirectly connected components" have been defined first by Verhoeff, cf. [Verhoeff85]. From definition 4.30, "computation interference hazard for indirect connection", follows the symmetry of ***NCIHDS*** .

property 4.31 *symmetry of NCIHDS*

For i/o-connectable components Γ and Δ,

$$\Gamma \; \textit{NCIHDS} \; \Delta = \Delta \; \textit{NCIHDS} \; \Gamma$$

end of property

Since indirect connections are used to model nonnegative delays, whereas direct connections are used to model zero delays only, we find the relations between "computation interference hazard for indirectly connected components" and "computation interference hazard for directly connected components" shown in property 4.32.

property 4.32

For i/o-connectable components Γ and Δ,

(i) $\Gamma \; \textit{NCIHADS} \; \Delta \Rightarrow \Gamma \; \textit{NCIHA} \; \Delta$

(ii) $\Gamma \; \textit{NCIHDS} \; \Delta \Rightarrow \Gamma \; \textit{NCIH} \; \Delta$

end of property

In general, implications from right to left in property 4.32 do not hold, see example 4.33.

example 4.33

We consider component Γ_2 that is defined by

$$\mathbf{i}\Gamma_2 \stackrel{\text{def}}{=} \varnothing, \qquad \mathbf{o}\Gamma_2 \stackrel{\text{def}}{=} \{a, b\}, \qquad \mathbf{t}(\mathbf{ptr}\,\Gamma_2) \stackrel{\text{def}}{=} \{\varepsilon, a, ab\},$$
$$\mathbf{o}\Delta_2 \stackrel{\text{def}}{=} \varnothing, \qquad \mathbf{i}\Delta_2 \stackrel{\text{def}}{=} \{a, b\}, \qquad \mathbf{t}(\mathbf{ptr}\,\Delta_2) \stackrel{\text{def}}{=} \{\varepsilon, a, ab\}.$$

We see that Γ_2 ***NCIHA*** Δ_2 , but not Γ_2 ***NCIHADS*** Δ_2 . Furthermore, we see that Γ_2 ***NCIH*** Δ_2 , but not Γ_2 ***NCIHDS*** Δ_2 .

end of example

In chapters 4, 5, and 6 we present more properties that show relations between ***NCIHA*** and ***NCIHADS*** and between ***NCIH*** and ***NCIHDS*** .

4.2.1 Delay-safe enclosure

We are interested in the communication behavior of a component, say Γ, that has an indirect connection with its environment, say Δ. In order to study this communication behavior and the communication between Γ and Δ, we introduce the notion *delay-safe enclosure* of a component. The delay-safe enclosure of component Γ is a component. It is denoted by **DSE** Γ. Using the delay-safe enclosure, we learn about the indirect connection of Γ and Δ by studying the direct connection of **DSE** Γ and **DSE** Δ, see figure 4.5.

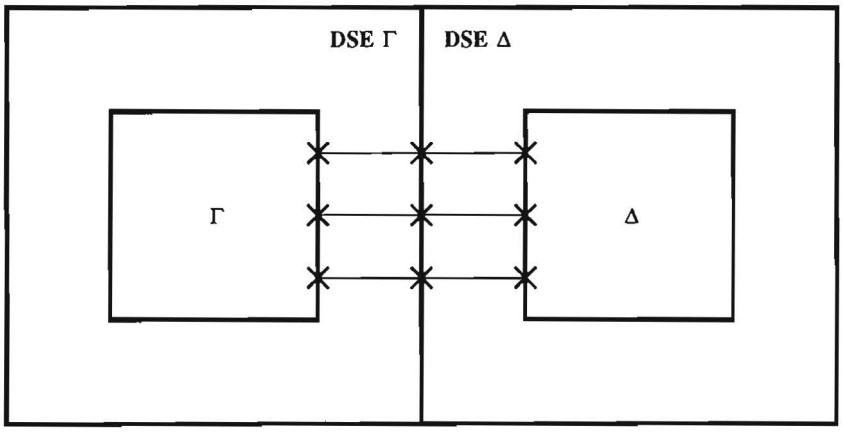

figure 4.5
Components Γ and Δ and their delay-safe enclosures.

We will define the delay-safe enclosure such that the indirectly connected components Γ and Δ communicate delay-safely and have absence of computation interference hazard, if and only if the directly connected components **DSE** Γ and **DSE** Δ have absence of computation interference hazard, see theorem 4.56.

The reflection of the delay-safe enclosure of a component, say Γ, can be interpreted as an environment of Γ that is able to communicate delay-safely with Γ, see figure 4.6.

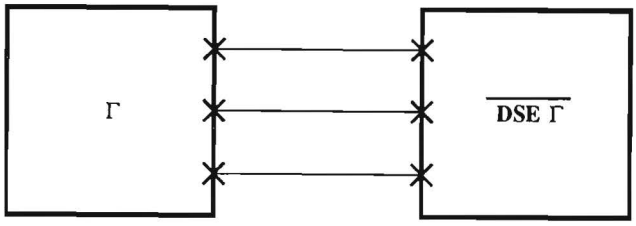

figure 4.6
Components Γ and $\overline{\mathbf{DSE}\,\Gamma}$.

For component Γ, component $\overline{\mathbf{DSE}\,\Gamma}$ is the maximal (w.r.t. trace structure inclusion) partner of Γ; when Γ and $\overline{\mathbf{DSE}\,\Gamma}$ are indirectly connected, they have no computation interference hazard, see definition 4.34.

definition 4.34 *delay-safe enclosure*

For component Γ, we define the *delay-safe enclosure* of Γ, denoted by $\mathbf{DSE}\,\Gamma$, as the maximal (w.r.t. trace structure inclusion) component such that

(i) $\mathbf{io}\,\Gamma = \mathbf{io}(\mathbf{DSE}\,\Gamma)$

(ii) $\Gamma\ \mathit{NCIHDS}\ \overline{\mathbf{DSE}\,\Gamma}$

(iii) $(\mathbf{A}\,a,t : a \in \mathbf{i}(\overline{\mathbf{DSE}\,\Gamma}) \wedge ta \in \mathbf{t}(\mathbf{ptr}(\overline{\mathbf{DSE}\,\Gamma})) : (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,ta))$

end of definition

The existence of the maximum in definition 4.34, "delay-safe enclosure", above follows from the "delay-safe enclosure" theorem 4.45. Requirement (iii) in definition 4.34, "delay-safe enclosure", restricts in the traces of $\overline{\mathbf{DSE}\,\Gamma}$ the occurrences of symbols $a \in \mathbf{i}(\overline{\mathbf{DSE}\,\Gamma})$ to those occurrences that are associated with the reception by $\overline{\mathbf{DSE}\,\Gamma}$ of commsigs that may have been sent by Γ. In requirement (iii) there is no need to quantify over symbols $a \in \mathbf{o}(\overline{\mathbf{DSE}\,\Gamma})$, since their occurrences in traces of $\mathbf{ptr}(\overline{\mathbf{DSE}\,\Gamma})$ are restricted by requirement (ii).

remark 4.35

Molnar introduced the metaphor "a component wrapped in a *Foam Rubber Wrapper*" for a component that communicates delay-safely, cf. [Schols85]. Readers that are familiar with this metaphor will recognize the delay-safe enclosure as its formalization.

end of remark

For component Γ we define trace structure $\mathbf{dse}\,\Gamma$. It will turn out that this is the trace structure of the delay-safe enclosure of Γ, see theorem 4.45.

definition 4.36 **dse**

For component Γ, we define trace structure $\mathbf{dse}\,\Gamma$ recursively by

(i) $\mathbf{a}(\mathbf{dse}\,\Gamma) \stackrel{\text{def}}{=} \mathbf{a}(\mathbf{ptr}\,\Gamma)$

(ii) $\varepsilon \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

(iii) for trace x and symbol a such that $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, $a \in \mathbf{o}\,\Gamma$, and $(\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge s\,\mathbf{c}_{\mathbf{io}\,\Gamma}\,x : \#_a s > \#_a x)$,

$$xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$$

(iv) for trace x and symbol a such that $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, $a \in \mathbf{i}\,\Gamma$, and $(\mathbf{A}\,s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\,\Gamma \wedge s\,\mathbf{c}_{\mathbf{io}\,\Gamma}\,xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$,

$$xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$$

(v) completeness axiom: $\mathbf{t}(\mathbf{dse}\,\Gamma)$ contains no elements that are not required by (ii), (iii), or (iv).

end of definition

In definition 4.34 (ii) absence of computation interference hazard between the indirectly connected Γ and $\overline{\mathbf{DSE}\,\Gamma}$ has been required. In definition 4.36, (iii) reflects that a component may produce any output whenever this output is enabled, and, hence, $\overline{\mathbf{DSE}\,\Gamma}$ accepts any input from Γ, whenever it receives this input. Furthermore, (iv) reflects that Γ accepts all inputs it might receive from $\overline{\mathbf{DSE}\,\Gamma}$, and, hence, $\overline{\mathbf{DSE}\,\Gamma}$ does not produce any output unless Γ is able to accept it. In addition to this, the quantification over input b in (iv) reflects that $\overline{\mathbf{DSE}\,\Gamma}$ may only produce an output when this will not prevent Γ from accepting all inputs that it might receive from $\overline{\mathbf{DSE}\,\Gamma}$, cf. example 4.37.

example 4.37

We consider component Γ_3, see figure 4.7.

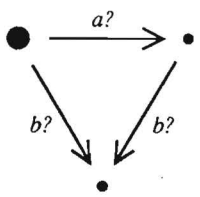

figure 4.7
State graph of component Γ_3.

Using definition 4.36, "**dse**", we find that $\mathbf{t}(\mathbf{dse}\,\Gamma_3) = \{\varepsilon, a, b\}$. Let symbols a and b be associated with commsig α_1 and β_1, respectively. We notice that trace ab is not a member of trace set $\mathbf{t}(\mathbf{dse}\,\Gamma_3)$, despite that Γ_3 will accept β_1. The reason for the absence of ab is that Γ_3 might receive β_1 first; hereafter Γ_3 will not accept α_1 any more.

end of example

We illustrate definition 4.36, "**dse**", by calculating **dse** for some components introduced in chapter 2.

example 4.38

We consider component Γ_w, see example 2.47. From definition 4.36, "**dse**", we conclude that $\mathbf{dse}\,\Gamma_w = \mathbf{ptr}\,\Gamma_w$, see figure 4.8.

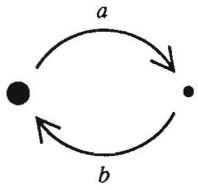

figure 4.8
State graph of trace structures $\mathbf{ptr}(\Gamma_w)$ and $\mathbf{dse}\,\Gamma_w$.

end of example

example 4.39

We consider component Γ_{af}, see example 2.49 and figure 4.9a. Using definition 4.36, "**dse**", we calculate $\mathbf{dse}\,\Gamma_{af}$, see figure 4.9b.

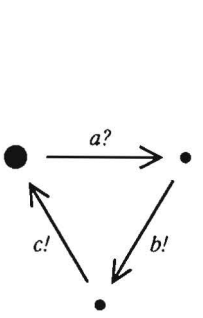

figure 4.9a

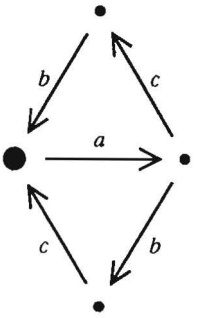

figure 4.9b

State graphs of component Γ_{af} (figure 4.9a) and trace structure $\mathbf{dse}\,\Gamma_{af}$ (figure 4.9b).

We notice that $\mathbf{dse}\,\Gamma_{af} = \mathbf{ptr}\,\Gamma_f$, see example 2.49

end of example

The following properties and lemmas are used in the proof of theorem 4.45, "delay-safe enclosure".

property 4.40

For component Γ,

(i) $\varepsilon \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

(ii) $\mathbf{dse}\,\Gamma$ is prefix-closed

end of property

property 4.41

For component Γ,

$$(\mathbf{A}\, s, b, x : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge sb\mathbf{c}_{\mathbf{io}\Gamma}x \wedge x \in \mathbf{t}(\mathbf{dse}\,\Gamma) : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$$

end of property

Property 4.42 will be used in theorem 4.45 to reflect that there is absence of computation interference hazard between indirectly connected components Γ and $\overline{\mathbf{DSE}\,\Gamma}$.

property 4.42

Given is component Γ. Let Δ be a component such that $\mathbf{io}\,\Delta = \mathbf{io}\,\overline{\Gamma}$ and $\mathbf{ptr}\,\Delta = \mathbf{dse}\,\Gamma$. Now,

$$\Gamma \; \textit{NCIHDS} \; \Delta$$

end of property

lemma 4.43

For component Γ,

$$(\mathbf{A}\, t : t \in \mathbf{t}(\mathbf{dse}\,\Gamma) : (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\, t))$$

end of lemma

lemma 4.44

For components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$ and $\overline{\Delta}\; \textit{NCIHADS}\; \Gamma$,

$$(\mathbf{A}\, s, t : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t \in \mathbf{t}(\mathbf{ptr}\,\Delta) \setminus \mathbf{t}(\mathbf{dse}\,\Gamma) : \neg (s\mathbf{c}_{\mathbf{io}\,\Gamma}\, t))$$

end of lemma

Now, we can link trace structure $\mathbf{dse}\,\Gamma$ to component $\mathbf{DSE}\,\Gamma$.

theorem 4.45 *delay-safe enclosure*

For component Γ,

$$\mathbf{ptr}(\mathbf{DSE}\,\Gamma) = \mathbf{dse}\,\Gamma$$

end of theorem

remark 4.46

The operator **dse** is equal to Verhoeff's operator ^, cf. [Verhoeff85]. We consider component Γ; in our terminology, Verhoeff considers all components Δ with $\Delta\; \textit{NCIHDS}\; \Gamma$ and $\mathbf{io}\,\Delta = \mathbf{io}\,\overline{\Gamma}$. He defines $\mathbf{dse}\,\Gamma$ as the union of the trace structures of the channels between each such a Δ and Γ. Our definition is constructive: starting from ε, every trace of $\mathbf{t}(\mathbf{dse}\,\Gamma)$ can be constructed in the way described in definition 4.36, "**dse**".

end of remark

remark 4.47

The following example illustrates that, for component Γ, $\mathbf{ptr}\,\Gamma$ and $\mathbf{dse}\,\Gamma$ are, in general, not ordered with respect to trace structure inclusion, see also [van der Veeken87]; Chen, Udding, and Verhoeff have defined a different, more complex order with respect to which $\mathbf{ptr}\,\Gamma$ and $\mathbf{dse}\,\Gamma$ are ordered, see [Chen – Udding – Verhoeff89].

end of remark

example 4.48

We consider component Γ_4, see figure 4.10.

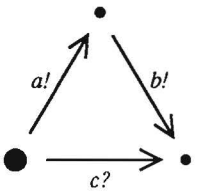

figure 4.10
State graph of component Γ_4.

Using definition 4.36, "**dse**", we calculate $\mathbf{dse}\,\Gamma_4$; $\mathbf{io}(\mathbf{DSE}\,\Gamma_4) = \mathbf{io}\,\Gamma_4$; trace set $\mathbf{t}(\mathbf{dse}\,\Gamma_4)$ is shown in figure 4.11.

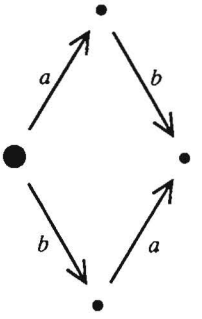

figure 4.11
State graph of trace set $\mathbf{t}(\mathbf{dse}\,\Gamma_4)$.

Let iobip F_4 be such that $F_4 = \mathbf{io}\,\Gamma_4$. From $abc_{F_4}b$ follows $b \in \mathbf{t}(\mathbf{dse}\,\Gamma_4)$, cf. definition 4.34 (iii). Analogously, from $abc_{F_4}ba$ follows $ba \in \mathbf{t}(\mathbf{dse}\,\Gamma_4)$. We see that the ordering of a and b is lost. We conclude that $\neg(\mathbf{dse}\,\Gamma_4 \subseteq \mathbf{ptr}\,\Gamma_4)$. Furthermore, from $ac_{F_4}c$, $a \in \mathbf{t}(\mathbf{ptr}\,\Gamma_4)$ and $ac \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_4)$ follows $c \notin \mathbf{t}(\mathbf{dse}\,\Gamma_4)$, cf. definition 4.36 (iv). We conclude that $\neg(\mathbf{ptr}\,\Gamma_4 \subseteq \mathbf{dse}\,\Gamma_4)$.

end of example

4.2.2 Properties of the delay-safe enclosure

In this subsection we present some properties of **DSE**. Of course, see theorem 4.45, "delay-safe enclosure", we also present some properties of **dse**.

lemma 4.49

For component Γ,

$$(\mathbf{A}\, t : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge (\mathbf{E}\, y : y \in \mathbf{t}(\mathbf{dse}\,\Gamma) : t\mathbf{c}_{\mathbf{io}\,\Gamma}\, y) : t \in \mathbf{t}(\mathbf{dse}\,\Gamma))$$

end of lemma

From lemma 4.49 we infer property 4.50.

property 4.50

For component Γ,

$$(\mathbf{A}\, t : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : (\mathbf{E}\, y : y \in \mathbf{t}(\mathbf{dse}\,\Gamma) : t\mathbf{c}_{\mathbf{io}\,\Gamma}\, y)) \Rightarrow \mathbf{ptr}\,\Gamma \subseteq \mathbf{dse}\,\Gamma$$

end of property

lemma 4.51

For component Γ,

$$(\mathbf{dse}\,\Gamma, \mathbf{ab}\,\Gamma) \in \mathbf{D}_4$$

end of lemma

lemma 4.52

For component Γ,

$$(\mathbf{ptr}\,\Gamma, \mathbf{ab}\,\Gamma) \in \mathbf{D}_4 = (\mathbf{dse}\,\Gamma = \mathbf{ptr}\,\Gamma)$$

end of lemma

From theorem 4.45, "delay-safe enclosure", lemma 4.51, and lemma 4.52 we infer property 4.53.

property 4.53 **DSE** *is idempotent*

For component Γ,

$$\mathbf{DSE}(\mathbf{DSE}\,\Gamma) = \mathbf{DSE}\,\Gamma$$

end of property

4.2.2.0 Computation interference hazard

In this subsection we present some properties about the delay-safe enclosure and computation interference hazard.

property 4.54

For i/o-connectable components Γ and Δ,

(i) $\Gamma\ \mathit{NCIHADS}\ \Delta = (\mathbf{DSE}\ \Gamma)\ \mathit{NCIHADS}\ \Delta$

(ii) $\Gamma\ \mathit{NCIHADS}\ \Delta = \Gamma\ \mathit{NCIHADS}\ (\mathbf{DSE}\ \Delta)$

(iii) $\Gamma\ \mathit{NCIHDS}\ \Delta = (\mathbf{DSE}\ \Gamma)\ \mathit{NCIHDS}\ \Delta$

end of property

The delay-safe enclosure enables us to express "(input) computation interference hazard when the communication is delay-safe" in terms of "(input) computation interference hazard". In order to do this we substitute one of the components by its delay-safe enclosure, see property 4.55 (iii).

property 4.55

For i/o-connectable components Γ and Δ,

(i) $\Gamma\ \mathit{NCIHADS}\ \Delta = (\mathbf{DSE}\ \Gamma)\ \mathit{NCIHA}\ \Delta$

(ii) $\Gamma\ \mathit{NCIHADS}\ \Delta = \Gamma\ \mathit{NCIHA}\ (\mathbf{DSE}\ \Delta)$

(iii) $\Gamma\ \mathit{NCIHDS}\ \Delta = (\mathbf{DSE}\ \Gamma)\ \mathit{NCIH}\ \Delta$

end of property

From property 4.54 (iii) and property 4.55 (iii) we conclude that the delay-safe enclosure has been defined such that the indirectly connected components Γ and Δ communicate delay-safely and have absence of computation interference hazard, if and only if the directly connected components $\mathbf{DSE}\ \Gamma$ and $\mathbf{DSE}\ \Delta$ have absence of computation interference hazard, cf. theorem 4.56.

theorem 4.56

For i/o-connectable components Γ and Δ,

$$\Gamma\ \mathit{NCIHDS}\ \Delta = (\mathbf{DSE}\ \Gamma)\ \mathit{NCIH}\ (\mathbf{DSE}\ \Delta)$$

end of theorem

4.2.2.1 Trace structure inclusion

In this subsection we present some examples that show some properties of trace structures **ptr** Γ, **dse** Γ, and **dsc** Γ, for component Γ.

remark 4.57

In general, the delay-safe enclosure is not monotonic w.r.t. trace structure inclusion, as is shown in example 4.58 and in example 4.81.

end of remark

example 4.58

We consider component Γ_5; the state graph of Γ_5 is given in figure 4.12.

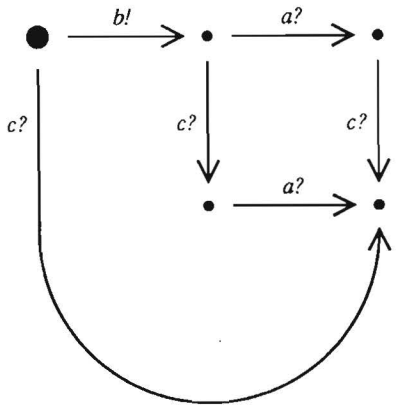

figure 4.12
State graph of component Γ_5.

The state graph of trace structure **dsc** Γ_5 is given in figure 4.13.

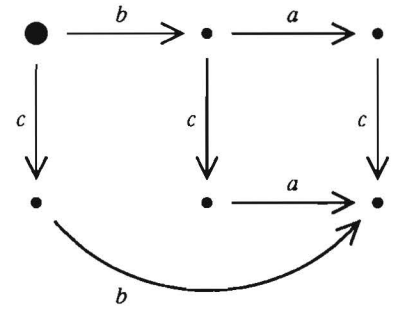

figure 4.13
State graph of trace structure **dsc** Γ_5.

The state graph of component $\mathbf{DSE}\,\Gamma_5$ is given in figure 4.14.

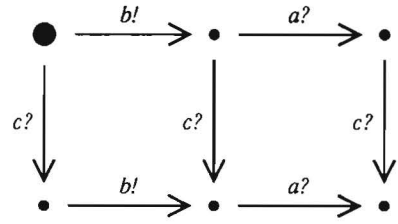

figure 4.14
State graph of component $\mathbf{DSE}\,\Gamma_5$.

We see that $cba \notin \mathbf{t}(\mathbf{dsc}\,\Gamma_5)$ and $cba \in \mathbf{t}(\mathbf{dse}\,\Gamma_5)$.

Let component Δ_5 be such that $\mathbf{io}\,\Delta_5 = \mathbf{io}\,\Gamma_5$ and $\mathbf{ptr}\,\Delta_5 = \mathbf{dsc}\,\Gamma_5$. From $(\mathbf{ptr}\,\Delta_5, \mathbf{ab}\Gamma_5) \in \mathbf{D}_4$ and lemma 4.52 it follows that $\mathbf{dse}\,\Delta_5 = \mathbf{ptr}\,\Delta_5$. Now, it can be seen that $\neg(\mathbf{dse}\,\Gamma_5 \subseteq \mathbf{dse}\,\Delta_5)$; nevertheless, $\mathbf{ptr}\,\Gamma_5 \subseteq \mathbf{ptr}\,\Delta_5$.

end of example

remark 4.59

In general, $\neg(\mathbf{dse}\,\Gamma \subseteq \mathbf{dsc}\,\Gamma)$, for component Γ, see example 4.58; however, there exist components Γ for which $\mathbf{dse}\,\Gamma \subset \mathbf{dsc}\,\Gamma$, see example 4.60.

end of remark

example 4.60

We consider component Γ_{lubc}, see example 2.50. The state graph of Γ_{lubc} is shown in figure 4.15. The state graph of trace structure $\mathbf{dsc}\,\Gamma_{lubc}$ is shown in figure 4.16.

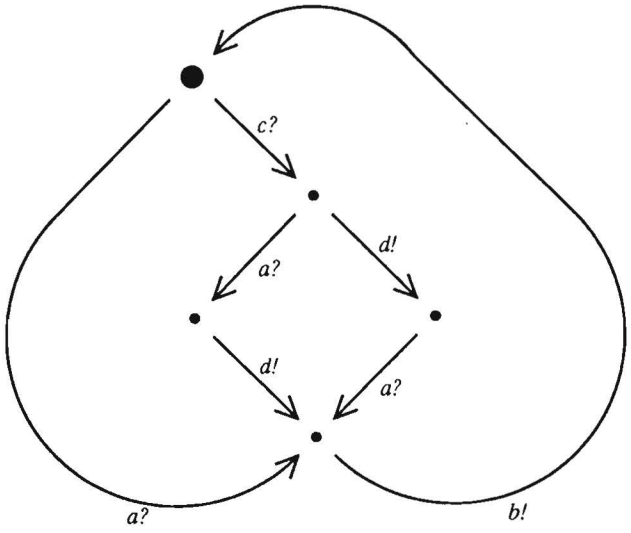

figure 4.15
State graph of component Γ_{lubc}.

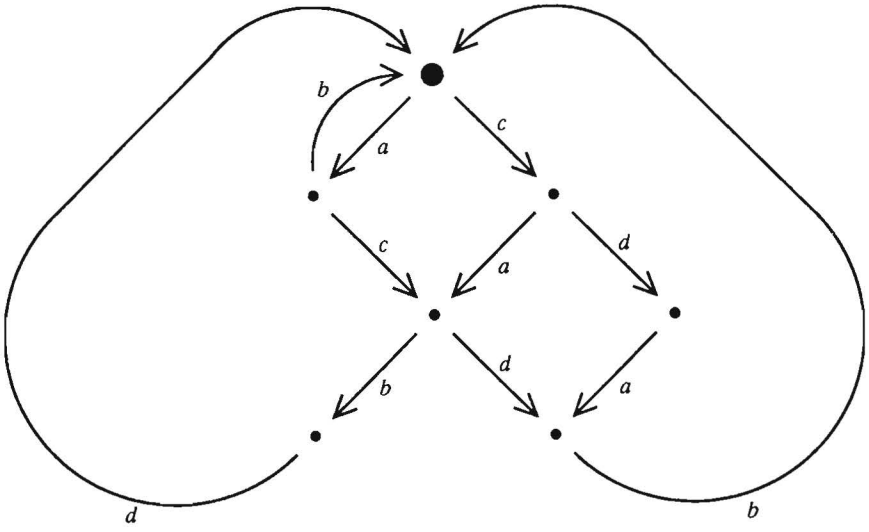

figure 4.16
State graph of trace structure $\mathbf{dsc}\,\Gamma_{lubc}$.

The state graph of component $\mathbf{DSE}\,\Gamma_{tubc}$ is shown in figure 4.17.

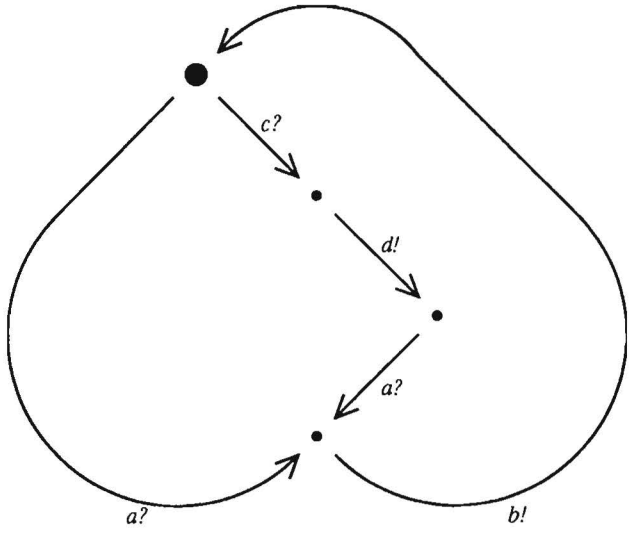

figure 4.17
State graph of component $\mathbf{DSE}\,\Gamma_{tubc}$.

Despite $ca \in \mathbf{t}(\mathbf{ptr}\,\Gamma_{tubc})$, ca is not an element of $\mathbf{t}(\mathbf{dse}\,\Gamma_{tubc})$, while $ac \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_{tubc})$, cf. definition 4.36 (iv). We notice that $\mathbf{dse}\,\Gamma_{tubc} \subset \mathbf{dsc}\,\Gamma_{tubc}$ (i.e. $\mathbf{dse}\,\Gamma_{tubc} \subseteq \mathbf{dsc}\,\Gamma_{tubc}$ and $\mathbf{dse}\,\Gamma_{tubc} \neq \mathbf{ptr}\,\Gamma_{tubc}$).

end of example

example 4.61

We consider component Γ_{bc}, see example 2.50. The state graph of component **DSE** Γ_{bc} is shown in figure 4.18.

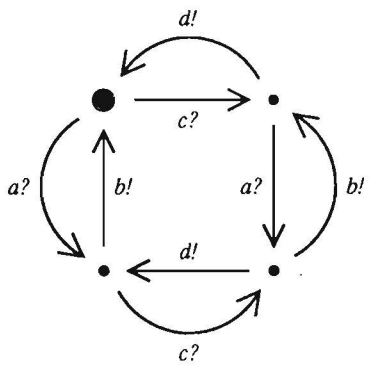

figure 4.18
State graph of component **DSE** Γ_{bc}.

We see that **DSE** $\Gamma_{bc} = \Gamma_{2w}$, see example 2.50. Furthermore, using definition 4.24, "**dsc** of component or channel", we find that **dse** Γ_{bc} = **dsc** Γ_{bc}.

end of example

4.2.2.2 Regularity and choice

In this subsection we present some examples to illustrate some properties of the delay-safe enclosure.

remark 4.62

Example 4.63 shows that the delay-safe enclosure does not preserve regularity.

end of remark

example 4.63 **DSE** *does not preserve regularity*

We consider component Γ_6, see figure 4.19.

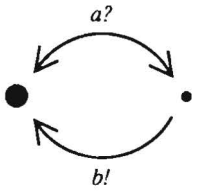

figure 4.19
State graph of component Γ_6.

Since the number of states of Γ_6 is finite, the state graph of component Γ_6 is regular, see subsection 1.4.2. Using definition 4.36, "**dse**", we infer that

$$(\mathbf{A}\, n : n \in \mathbb{N} : (a^n b^n \in \mathbf{t}(\mathbf{dse}\,\Gamma_6)) \wedge (a^n b^{n+1} \notin \mathbf{t}(\mathbf{dse}\,\Gamma_6))),$$

where a^n denotes the trace that consists of n symbols that are all equal to a. We notice that the number of states of $\mathbf{dse}\,\Gamma_6$ is infinite. Using theorem 4.45, "delay-safe enclosure", we conclude that the state graph of component $\mathbf{DSE}\,\Gamma_6$ is not regular.

Analogously, the regularity of the state graphs of components Γ_{or} and Γ_{and}, see example 2.51, and Γ_{maj}, see example 2.52, is not preserved by **DSE**.

end of example

remark 4.64

Example 4.65 shows that the delay-safe enclosure does not preserve "absence of choice between outputs".

end of remark

example 4.65

We consider component Γ_7; the state graph of Γ_7 is shown in figure 4.20.

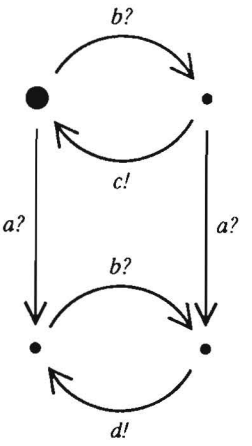

figure 4.20
State graph of component Γ_7.

The state graph of component **DSE** Γ_7 is shown in figure 4.21.

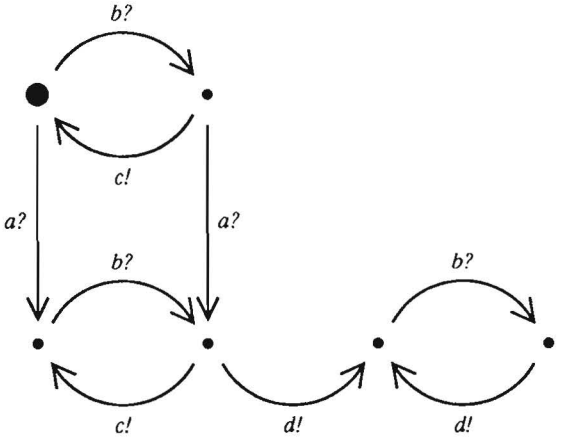

figure 4.21
State graph of component **DSE** Γ_7.

Whereas in Γ_7 "no outputs disable each other", in **DSE** Γ_7 this does not hold: $abc \in \mathbf{t}(\mathbf{dse}\,\Gamma_7)$ and $abd \in \mathbf{t}(\mathbf{dse}\,\Gamma_7)$, but $abcd \notin \mathbf{t}(\mathbf{dse}\,\Gamma_7)$ and $abdc \notin \mathbf{t}(\mathbf{dse}\,\Gamma_7)$.

In example 4.67 we will refer to $\mathbf{dsc}\,\Gamma_7$; for this reason we present this trace structure in figure 4.22.

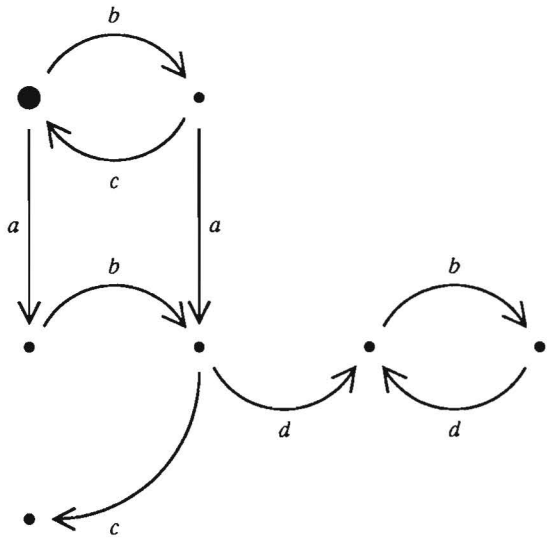

figure 4.22
State graph of trace structure $\mathbf{dsc}\,\Gamma_7$.

Notice that *abc* has no successor in $\mathbf{t}(\mathbf{dsc}\,\Gamma_7)$.

end of example

remark 4.66

In example 4.67 we present two components that have the same **dsc** but different **dse**.

end of remark

example 4.67

We consider components Γ_8 and Δ_8; Γ_8 is equal to component Γ_7 in example 4.65. Δ_8 has two inputs a and b, and two outputs c and d. The state graph of component Δ_8 is shown in figure 4.23.

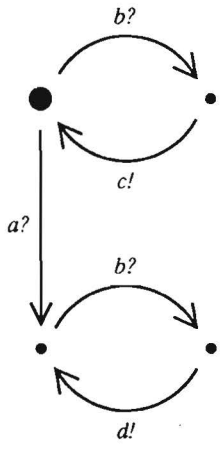

figure 4.23
State graph of component Δ_8.

Notice that the state graph of Δ_8 is almost equal to the state graph of component Γ_8. From definition 4.24, "**dsc** of component or channel", we infer that $\mathbf{dsc}\,\Delta_8 = \mathbf{dsc}\,\Gamma_8$, see figure 4.22.

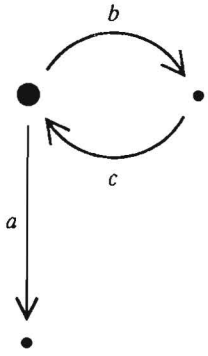

figure 4.24
State graph of $\mathbf{t}(\mathbf{dse}\,\Delta_8)$.

In figure 4.24 we show the state graph of trace set $\mathbf{t}(\mathbf{dse}\,\Delta_8)$. Let iobip F_8 be such that $\mathbf{io}\,F_8 = \mathbf{io}\,\Gamma_8$. Using definition 4.36 (iv), we infer from $bca\mathbf{c}_{F_8}ab$, $bc \in \mathbf{t}(\mathbf{ptr}\,\Delta_8)$, and $bca \notin \mathbf{t}(\mathbf{ptr}\,\Delta_8)$, that $ab \notin \mathbf{t}(\mathbf{dse}\,\Delta_8)$. Using that $ab \in \mathbf{t}(\mathbf{dse}\,\Gamma_8)$, we conclude that $\mathbf{dse}\,\Gamma_8 \neq \mathbf{dse}\,\Delta_8$, whereas $\mathbf{dsc}\,\Gamma_8 = \mathbf{dsc}\,\Delta_8$.

end of example

4.2.3 Behavior of delay-safely communicating components

In remark 4.47 we noticed that, in general, **ptr** Γ and **dse** Γ are not ordered with respect to trace structure inclusion. This is due to to the fact that the boundaries at which components Γ and **DSE** Γ are interpreted do not coincide, cf. figure 4.5. In this subsection we are interested in the impact of delay-safe communication on the communication behavior of a component. We define the *maximal communication behavior of a component that communicates delay-safely*, i.e. the maximal communication behavior of the component at the commports of the component when the component has an indirect connection with its environment and there is absence of computation interference hazard between them. The "maximal communication behavior of component Γ that communicates delay-safely" is a component. It is denoted by **CBDS** Γ. Components Γ and **CBDS** Γ are interpreted at the same boundary, see figures 4.25 and 4.26.

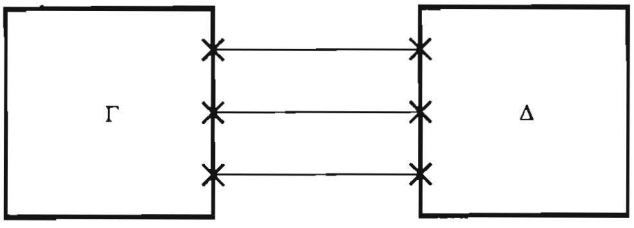

figure 4.25
Components Γ and Δ.

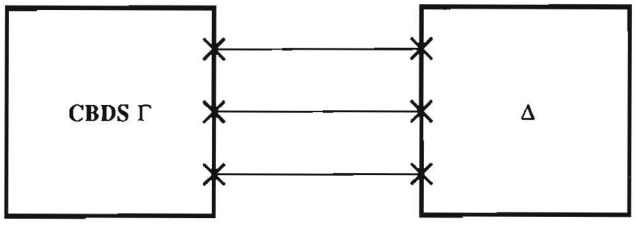

figure 4.26
Components **CBDS** Γ and Δ.

We consider the indirectly connected components Γ and Δ that communicate delay-safely and have absence of computation interference hazard. Let, at some moment, trace t be associated with a comminstorder of Γ and trace u be associated with a comminstorder of Δ such that t and u are consistent with our causality notion. We will define **CBDS** such that $t \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma))$, $u \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Delta))$, and $t\mathbf{c}_{\mathbf{io}\,\Gamma}u$; this is formally expressed by theorem 4.77. Furthermore, for every trace $t \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma))$ there exist a component Δ (e.g. $\overline{\mathbf{DSE}\,\Gamma}$) and a trace $u \in \mathbf{t}(\mathbf{ptr}\,\Delta)$ such that Γ and Δ communicate delay-safely without computation interference hazard, and, at some moment, t is associated with the order of comminsts at the commports of Γ, u is associated with the order of comminsts at the commports of Δ, and t and u are consistent with our causality notion; this is formally expressed by property 4.78.

definition 4.68 *maximal communication behavior for delay-safe communication*

For component Γ, we define the *maximal communication behavior of* Γ *when* Γ *communicates delay-safely*, denoted by $\mathbf{CBDS}\,\Gamma$, as the maximal (w.r.t. trace structure inclusion) component such that

(i) $\mathbf{io}(\mathbf{CBDS}\,\Gamma) = \mathbf{io}\,\Gamma$

(ii) $\mathbf{ptr}(\mathbf{CBDS}\,\Gamma) \subseteq \mathbf{ptr}\,\Gamma$

(iii) $(\mathbf{A}\,a, s : a \in \mathbf{i}\Gamma \wedge sa \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma)) : (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{dse}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\,\Gamma}t))$

end of definition

The existence of the maximum in definition 4.68, "maximal communication behavior for delay-safe communication", follows from theorem 4.74. In requirement (iii) of definition 4.68 we restrict the communication behavior of $\mathbf{CBDS}\,\Gamma$ by eliminating traces that are not composable under iobip $\mathbf{io}\Gamma$ with any trace of $\mathbf{dse}\,\Gamma$. In requirement (iii) we do not quantify over symbols $a \in \mathbf{o}\Gamma$, since there is no way to prevent a component from sending commsigs, cf. subsections 2.2.3, 2.1.4, and 2.1.3.

lemma 4.69

For component Γ,

$$(\mathbf{CBDS}\,\Gamma)\ \mathit{NCIHDS}\ \overline{\mathbf{DSE}\,\Gamma}$$

end of lemma

From lemma 4.69 we conclude that there is no need to require that $(\mathbf{CBDS}\,\Gamma)\ \mathit{NCIHDS}\ \overline{\mathbf{DSE}\,\Gamma}$ in definition 4.68, "maximal communication behavior for delay-safe communication".

For component Γ we define trace structure $\mathbf{cbds}\,\Gamma$. It will turn out that this is the trace structure of $\mathbf{CBDS}\,\Gamma$, see theorem 4.74.

definition 4.70 **cbds**

For component Γ trace structure $\mathbf{cbds}\,\Gamma$ is defined by

$$\mathbf{cbds}\,\Gamma \stackrel{\text{def}}{=} <\mathbf{a}\Gamma, \{t, u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\Gamma} u \wedge u \in \mathbf{t}(\mathbf{dse}\,\Gamma) : t\} >$$

end of definition

The definition above reflects that only the traces in $\mathbf{t}(\mathbf{ptr}\,\Gamma)$, that are composable with some trace in $\mathbf{t}(\mathbf{dse}\,\Gamma)$, are associated with the maximal communication behavior of component Γ when Γ communicates delay-safely. For an appreciation of this definition we refer to theorem 4.74. Using definition 4.36, "**dse**", we derive the following property.

property 4.71

For component Γ, trace t, and symbol a,

(i) for $a \in \mathbf{o}\Gamma$, $\quad (t \in \mathbf{t}(\mathbf{cbds}\,\Gamma) \wedge ta \in \mathbf{t}(\mathbf{ptr}\,\Gamma)) = ta \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$

(ii) for $a \in \mathbf{i}\Gamma$, $\quad (t \in \mathbf{t}(\mathbf{cbds}\,\Gamma) \wedge ta \in \mathbf{t}(\mathbf{dse}\,\Gamma)) = ta \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$

end of property

In property 4.71(ii) $ta \in \mathbf{t}(\mathbf{cbds}\,\Gamma) \Rightarrow ta \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ follows from lemma 4.49; from lemma 4.49 we also infer property 4.72.

property 4.72

For component Γ,

$$\mathbf{cbds}\,\Gamma = \mathbf{ptr}\,\Gamma \cap \mathbf{dse}\,\Gamma$$

end of property

From the nonemptiness and prefix-closedness of **ptr** and **dse** we infer the nonemptiness and prefix-closedness of **cbds**.

property 4.73

For component Γ,

$\mathbf{cbds}\,\Gamma$ is nonempty and prefix-closed.

end of property

Now, we can link trace structure $\mathbf{cbds}\,\Gamma$ to component $\mathbf{CBDS}\,\Gamma$.

theorem 4.74 *maximal communication behavior for delay-safe communication*

For component Γ,

$$\mathbf{ptr}(\mathbf{CBDS}\,\Gamma) = \mathbf{cbds}\,\Gamma$$

end of theorem

example 4.75

We consider component Γ_4 of example 4.48, see figure 4.10. $\mathbf{io}(\mathbf{CBDS}\,\Gamma_4) = \mathbf{io}\,\Gamma_4$. Trace set $\mathbf{t}(\mathbf{dse}\,\Gamma_4)$ is shown in figure 4.11.

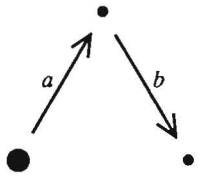

figure 4.27
State graph of trace set $\mathbf{t}(\mathbf{cbds}\,\Gamma_4)$.

In figure 4.27 the state graph of trace set $\mathbf{t}(\mathbf{cbds}\,\Gamma_4)$ is shown.

end of example

example 4.76

From definition 4.70, "**cbds**", we derive that $\mathbf{cbds}\,\Gamma_{or} = \mathbf{ptr}\,\Gamma_{or}$, $\mathbf{cbds}\,\Gamma_{and} = \mathbf{ptr}\,\Gamma_{and}$, $\mathbf{cbds}\,\Gamma_{bc} = \mathbf{ptr}\,\Gamma_{bc}$, $\mathbf{cbds}\,\Gamma_{maj} = \mathbf{ptr}\,\Gamma_{maj}$, and $\mathbf{cbds}\,\Gamma_{af} = \mathbf{ptr}\,\Gamma_{af}$; however, $\mathbf{cbds}\,\Gamma_{lubc} = \mathbf{dse}\,\Gamma_{lubc}$.

end of example

We now present some properties of **CBDS** announced in the introduction of this subsection. Theorem 4.77 expresses that $\mathbf{CBDS}\,\Gamma$ is not too small.

theorem 4.77

For components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\overline{\Delta}$,

$$\begin{aligned}&\Gamma\ \textit{NCIHDS}\ \Delta\\ \Rightarrow\ &(\mathbf{A}\,t,u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge tc_{\mathbf{io}\,\Gamma}\,u \wedge u \in \mathbf{t}(\mathbf{ptr}\,\Delta)\\ &\qquad : t \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma)) \wedge u \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Delta))\\ &)\end{aligned}$$

end of theorem

Property 4.78 expresses that $\mathbf{CBDS}\,\Gamma$ is not too large.

property 4.78

For component Γ,

$$(\mathbf{A}\,t : t \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma)) : (\mathbf{E}\,u : u \in \mathbf{t}(\mathbf{ptr}(\overline{\mathbf{DSE}\,\Gamma})) : tc_{\mathbf{io}\,\Gamma}\,u))$$

end of property

From lemma 4.43 and definition 4.36, "**dse**", we infer property 4.79; it is used in theorem 4.80.

property 4.79

For components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$ and $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$,

$$(s \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge t \in (\mathbf{o}\,\Gamma)^* \wedge st \in \mathbf{t}(\mathbf{dse}\,\Delta)) \Rightarrow st \in \mathbf{t}(\mathbf{dse}\,\Gamma)$$

end of property

Theorem 4.80 expresses that the traces that have been left out when reducing $\mathbf{ptr}\,\Gamma$ to $\mathbf{cbds}\,\Gamma$ do not play a role for a component that communicates delay-safely.

theorem 4.80

For components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$, $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$, and $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$,

$$\mathbf{dse}\,\Gamma = \mathbf{dse}\,\Delta$$

end of theorem

In theorem 4.80 we have proven that $\mathbf{dse}\,\Gamma = \mathbf{dse}\,\Delta$, for components Γ and Δ such that $(\mathbf{ptr}\,\Gamma \cap \mathbf{dse}\,\Gamma) \subseteq \mathbf{ptr}\,\Delta$ and $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$; in example 4.81 we show that this, in general, does not hold for components Γ and Δ such that $\mathbf{ptr}\,\Gamma \subset \mathbf{ptr}\,\Delta$ and $\mathbf{ptr}\,\Delta \subseteq (\mathbf{ptr}\,\Gamma \cup \mathbf{dse}\,\Gamma)$.

example 4.81

We consider components Γ_9 and Δ_9, see figure 4.28.

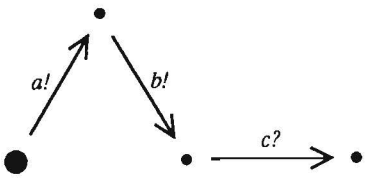

figure 4.28a

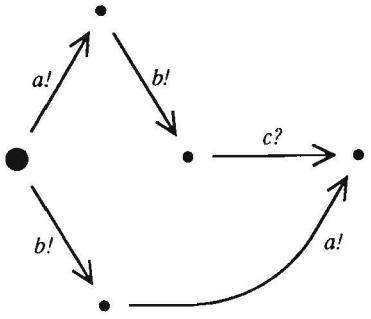

figure 4.28b

State graphs of components Γ_9 (figure 4.28a) and Δ_9 (figure 4.28b).

The delay-safe enclosures of Γ_9 and Δ_9 are given in figure 4.29; of course, $\mathbf{io}(\mathbf{DSE}\,\Delta_9) = \mathbf{io}\,\Delta_9$.

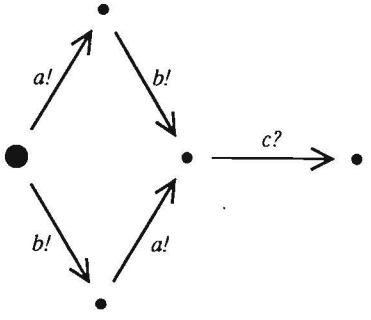

figure 4.29a

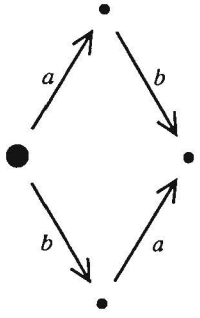

figure 4.29b

State graphs of component $\mathbf{DSE}\,\Gamma_9$ (fig. 4.29a) and trace set $\mathbf{t}(\mathbf{dse}\,\Delta_9)$ (fig. 4.29b).

We see that $\mathbf{ptr}\,\Gamma_9 \subset \mathbf{ptr}\,\Delta_9$ and $\mathbf{ptr}\,\Delta_9 \subseteq \mathbf{dse}\,\Gamma_9$, but $\neg(\mathbf{dse}\,\Gamma_9 = \mathbf{dse}\,\Delta_9)$.

end of example

lemma 4.82 **CBDS** *is idempotent*

For component Γ,

$$\mathbf{CBDS}(\mathbf{CBDS}\,\Gamma) = \mathbf{CBDS}\,\Gamma$$

end of lemma

property 4.83

For i/o-connectable components Γ and Δ,

(i) Γ ***NCIHADS*** $\Delta = (\mathbf{CBDS}\,\Gamma)$ ***NCIHADS*** Δ

(ii) Γ ***NCIHADS*** $\Delta = \Gamma$ ***NCIHADS*** $(\mathbf{CBDS}\,\Delta)$

(iii) Γ ***NCIHDS*** $\Delta = (\mathbf{CBDS}\,\Gamma)$ ***NCIHDS*** Δ

end of property

4.2.4 Impact of delay-safe communication on components

We consider a component Γ that communicates delay-safely, see figure 4.30.

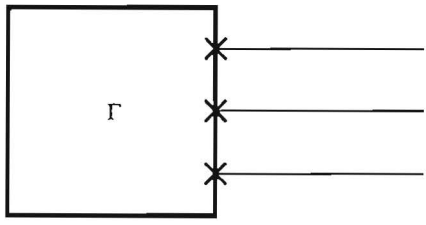

figure 4.30
Component Γ communicating delay-safely.

Now, the allowed communication behavior of Γ is restricted. We notice that in $\mathbf{CBDS}\,\Gamma$:

- no input comminst enables an input comminst, and
- no output comminst disables an input comminst.

lemma 4.84

For component Γ, no input comminst enables an input comminst in $\mathbf{CBDS}\,\Gamma$.

end of lemma

lemma 4.85

For component Γ, no output comminst disables an input comminst in $\mathbf{CBDS}\,\Gamma$.

end of lemma

remark 4.86

Example 4.87 illustrates that the lemmas above are the only relations between comminsts as far as enabling and disabling are concerned. In the six other cases no such lemmas can be derived: the existence of such lemmas is disproved by example 4.87.

end of remark

example 4.87

In the communication behavior of a delay-safely communicating component

(i) an input comminst may disable an input comminst,

(ii) an input comminst may disable an output comminst,

(iii) an input comminst may enable an output comminst,

(iv) an output comminst may enable an input comminst,

(v) an output comminst may disable an output comminst,

(vi) an output comminst may enable an output comminst.

Table 4.31 lists components that illustrate the above.

	$\mathbf{i}\Gamma$	$\mathbf{o}\Gamma$	$\mathbf{t}(\mathbf{ptr}\,\Gamma)$	$\mathbf{t}(\mathbf{dse}\,\Gamma)$	$\mathbf{t}(\mathbf{cbds}\,\Gamma)$
(i)	$\{a,b\}$	$\{c\}$	$\mathbf{pref}(\{ac,bc\}^*)$	$\mathbf{pref}(\{ac,bc\}^*)$	$\mathbf{pref}(\{ac,bc\}^*)$
(ii)	$\{a\}$	$\{b\}$	$\mathbf{pref}\{a,ba\}$	$\mathbf{pref}\{ab,ba\}$	$\mathbf{pref}\{a,ba\}$
(iii)	$\{a\}$	$\{b\}$	$\mathbf{pref}(\{ab\}^*)$	$\mathbf{pref}(\{ab\}^*)$	$\mathbf{pref}(\{ab\}^*)$
(iv)	$\{b\}$	$\{a\}$	$\mathbf{pref}(\{ab\}^*)$	$\mathbf{pref}(\{ab\}^*)$	$\mathbf{pref}(\{ab\}^*)$
(v)	$\{c\}$	$\{a,b\}$	$\mathbf{pref}(\{ac,bc\}^*)$	$\mathbf{pref}(\{ac,bc\}^*)$	$\mathbf{pref}(\{ac,bc\}^*)$
(vi)	$\{c\}$	$\{a,b\}$	$\mathbf{pref}(\{abc\}^*)$	$\mathbf{pref}(\{abc,bac\}^*)$	$\mathbf{pref}(\{abc\}^*)$

table 4.31

Example iobips and trace sets for components Γ, $\mathbf{DSE}\,\Gamma$, and $\mathbf{CBDS}\,\Gamma$.

end of example

remark 4.88

From example 4.89 we conclude that lemma 4.84 and lemma 4.85 are not sufficient to characterize **CBDS**.

end of remark

example 4.89

We consider component Γ_{10} that is defined by

$$\mathbf{i}\Gamma_{10} \stackrel{\text{def}}{=} \{a, c\},$$
$$\mathbf{o}\Gamma_{10} \stackrel{\text{def}}{=} \{b\}, \text{ and}$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma_{10}) \stackrel{\text{def}}{=} \mathbf{pref}\{abc, ba\}.$$

We see:

$$\mathbf{t}(\mathbf{dse}\,\Gamma_{10}) = \mathbf{pref}\{ab, ba\}, \text{ and}$$
$$\mathbf{t}(\mathbf{cbds}\,\Gamma_{10}) = \mathbf{pref}\{ab, ba\}.$$

For component Γ_{10} no input comminst enables an input comminst and no output comminst disables an input comminst; nevertheless, $\mathbf{ptr}\,\Gamma_{10} \neq \mathbf{cbds}\,\Gamma_{10}$.

end of example

4.2.5 'Off-the-shelf' mechanisms

In subsection 4.2.1 we have defined the operator **dse** for components. For component Γ, $\mathbf{dse}\,\Gamma$ is the trace structure of the delay-safe enclosure of Γ. Using **dse** we have defined the operator **cbds** in subsection 4.2.3; $\mathbf{cbds}\,\Gamma$ is the trace structure of the maximal communication behavior of Γ, if Γ communicates delay-safely. Let component Γ model some 'off-the-shelf' mechanism; now, trace structure $\mathbf{cbds}\,\Gamma$ can be used as a label that can be attached to such a mechanism to show its maximal delay-safe communication behavior.

Suppose that we have such a mechanism. From its specification we can tell the iobip of the component that models this mechanism. If there are no explicit timing requirements, we are able to derive the trace structure of this component from the specification of the mechanism. If there are explicit timing requirements (e.g. signal *II* may not happen until at least 3 microseconds after signal *I* has happened), we have to separate them from '(the rest of) the communication behavior of the mechanism'. Adding delay elements may be effective. An alternative is to add clock signals to indicate when the timing requirements have been met. In both cases we create a new mechanism from which the explicit timing requirements have been separated. Again, we are able to derive the trace structure of the component from the specification of the (new) mechanism. In this way, the component models the "new mechanism from which the explicit timing conditions have been separated".

After we have determined the trace structure of the component that models the mechanism, we calculate the **cbds** of this component. If this **cbds** is too restricted, we conclude that we do not want to require that the communication between this component and its environment is delay-safe. We might choose a mixed connection between this component and its environment, cf. partial delay-safety in chapter 6; or we might create a new mechanism, e.g. by adding some clock signals in the way described above.

5

Communicating delay-insensitively

In this chapter we deal with "absence of transmission interference hazard" within the context of delay-safe communication; as a consequence we are only concerned here with *indirect connections*. We consider a component and its environment that have a closed connection. We study the communication in the channel between this component and its environment. If a channel is delay-safe and there is no transmission interference hazard, we say that the channel is *delay-insensitive*.

In subsection 5.1.0.1 we present **DIE**, i.e. the delay-insensitive enclosure of a di-initializable (see subsection 5.1.0.0) component. For di-initializable component Γ, component $\overline{\mathbf{DIE}\,\Gamma}$ is the maximal (w.r.t. trace structure inclusion) partner of Γ. When Γ and $\overline{\mathbf{DIE}\,\Gamma}$ are indirectly connected, they have no computation interference hazard and there is no transmission interference hazard in the communication between them. In subsection 5.1.0.3 we present **CBDI**, i.e. the communication behavior of a delay-insensitively communicating di-initializable component. The maximal (w.r.t. trace structure inclusion) communication behavior of a di-initializable component, say Γ, that communicates delay-insensitively without computation interference hazard equals trace structure $\mathbf{cbdi}\,\Gamma$ ($\mathbf{cbdi}\,\Gamma \subseteq \mathbf{ptr}\,\Gamma$). This means that Γ behaves in that case like component $\mathbf{CBDI}\,\Gamma$ ($\mathbf{io}(\mathbf{CBDI}\,\Gamma) = \mathbf{io}\,\Gamma$ and $\mathbf{ptr}(\mathbf{CBDI}\,\Gamma) = \mathbf{cbdi}\,\Gamma$).

5.0 Communication in channels

We define class $\mathbf{C}_4$ in order to formalize "delay-insensitive channel". *Absence of transmission interference hazard* is characterized by:

no two signals are permitted to interfere with each other.

remark 5.0

Within the context of delay-safe communication absence of transmission interference hazard is equal to:

no commsig is sent from a commport before all commsigs, that previously have been sent from that commport, have been received.

end of remark

definition 5.1 $\mathbf{C}_4$

For trace structure T and alphbip D, the pair (T,D) is an element of $\mathbf{C}_4$ if and only if

$$(T,D) \in \mathbf{D}_4$$
$$\wedge\, (\mathbf{A}\, s,t,a : s \in (\mathbf{a}T)^* \wedge t \in (\mathbf{a}T)^* \wedge a \in \mathbf{a}T \wedge sata \in \mathbf{t}T :\ I(t \restriction \mathbf{opa}(a,D)) > 0)$$

end of definition

That the communication is delay-safe is reflected by the first conjunct in definition 5.1, "$\mathbf{C}_4$", cf. definition 4.19, "delay-safe channel". The second conjunct reflects, given that the communication is delay-safe, absence of transmission interference hazard: no commsig may propagate from a commport of the component at 'one end of the channel' to a commport of the component at 'the other end of the channel', unless all commsigs that have previously propagated between these commports have been received. As a consequence, using that the communication is delay-safe, no commsig must be sent from a commport unless all commsigs that have previously been sent from this commport have been received. Since the connection between the component and its environment is closed, the only way in which the component that sends these commsigs is able to know that a commsig has been received, is by receiving one or more commsigs that travel in the opposite direction.

Class $\mathbf{C}_4$ has been called the "delay-insensitive class" by Udding, cf. subsection 7.0.1. Definition 5.1, "$\mathbf{C}_4$", differs from Udding's original definition, cf. [Udding84]; in theorem 5.2, "$\mathbf{C}_4$", we prove that these definitions are equivalent. Udding's definition is simpler from a formal point of view; we believe that our definition is closer to our intuitive notion "absence of transmission interference hazard".

theorem 5.2 $\mathbf{C}_4$

For trace structure T and alphbip D,

$$(T,D)\in\mathbf{C}_4 \;=\; (T,D)\in\mathbf{D}_4 \wedge (\mathbf{A}\,s,a : s\in(\mathbf{a}T)^* \wedge a\in\mathbf{a}T : saa\notin\mathbf{t}T)$$

end of theorem

Notice, that in the proof of theorem 5.2, "$\mathbf{C}_4$", we need that the communication is delay-safe in order to prove that absence of transmission interference hazard is equal to Udding's requirement, viz. $(\mathbf{A}\,s,a : s\in(\mathbf{a}T)^* \wedge a\in\mathbf{a}T : saa\notin\mathbf{t}T)$. Using class $\mathbf{C}_4$, we define the notion "delay-insensitive channel".

definition 5.3 *delay-insensitive channel*

For channel Θ, we say that Θ *is delay-insensitive* if and only if

$$(\mathbf{ptr}\,\Theta,\mathbf{ab}\Theta)\in\mathbf{C}_4$$

end of definition

We deal with absence of transmission interference hazard as a property that may or may not hold for the communication in a delay-safe channel. Udding, cf. [Udding84], and Ebergen, cf. [Ebergen87], however, take delay-insensitivity as their starting point.

We do not define, for trace structure T and alphbip D, the smallest (w.r.t. trace structure inclusion) trace structure X such that $T\subseteq X$ and $(X,D)\in\mathbf{C}_4$: in general, such a trace structure X does not exist, see example 5.4. Furthermore, if such an X exists, then $X=\mathbf{dsc}(T,D)$, see definition 4.20, "**dsc**".

example 5.4

Component Γ_0 is defined by:

$$\mathbf{o}\Gamma_0 \stackrel{\text{def}}{=} \{a\}$$
$$\mathbf{i}\Gamma_0 \stackrel{\text{def}}{=} \{b\}$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma_0) \stackrel{\text{def}}{=} \mathbf{pref}\,\{baa\}$$

From theorem 5.2, "$\mathbf{C}_4$", we conclude that $(\mathbf{A}\,X : \mathbf{t}(\mathbf{ptr}\,\Gamma_0)\subseteq\mathbf{t}X : (X,\mathbf{ab}\Gamma_0)\notin\mathbf{C}_4)$.

end of example

5.1 Communication behavior of components

In this section we focus our attention on the communication behavior of a component that communicates via a delay-insensitive channel. In chapter 4 we have proven that delay-safe communication restricts the communication behavior of a component, when absence of computation interference hazard is the correctness concern. We shall show here that the additional correctness concern, viz. absence of transmission interference hazard, gives an additional restriction on the communication behavior of a component.

Throughout the remainder of this section we will only consider components that communicate via a delay-safe channel.

5.1.0 Transformation into computation interference hazard

In this subsection we use the transformation technique presented in subsection 3.3.0 to transform "transmission interference hazard" into "computation interference hazard", see subsection 5.1.0.1. We recall that there is an initial problem when this technique is applied. We deal with this problem in subsection 5.1.0.0.

5.1.0.0 Initializability

In subsection 4.2.4 we studied the restriction imposed by delay-safe communication on the communication behavior of components. We were able to calculate the maximal communication behavior of every component that communicates via a delay-safe channel. In this section we deal with the restriction imposed by "delay-insensitive communication" on the communication behavior of components. It turns out that some components are not able to communicate via a delay-insensitive channel: they may initially 'produce transmission interference' before the environment is able to control them. Such components are said to be not *di-initializable*.

definition 5.5 *di-initializable*

Component Γ is *di-initializable* if and only if

$$(\mathbf{A}\, s, t, a : s \in (\mathbf{a}\Gamma)^* \wedge t \in (\mathbf{a}\Gamma)^* \wedge a \in \mathbf{o}\Gamma \wedge sata \in \mathbf{t}(\mathbf{dse}\,\Gamma) : I(st \restriction \mathbf{opa}(a, \mathbf{ab}\Gamma)) > 0)$$

end of definition

We consider the condition that is used to define that a component is di-initializable, see definition 5.5. This condition looks very much like the second conjunct in definition 5.1, "$\mathbf{C}_4$". The restriction $sata \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ is included, because we assume that component Γ communicates via a delay-safe channel and because we are concerned with "absence of transmission interference hazard" only in the context of delay-safe communication. The restriction $a \in \mathbf{o}\Gamma$ is included, because Γ 'produces' the commsigs sent at its output commports, whereas the commsigs received at its input commports are 'produced' by some other component. The requirement $I(t \restriction \mathbf{opa}(a,D)) > 0$ is weakened to $I(st \restriction \mathbf{opa}(a,D)) > 0$, since we are only concerned with absence of transmission interference hazard in the initial part of the communication behavior of Γ. Notice that in definition 5.5 $\mathbf{opa}(a, \mathbf{ab}\Gamma) = \mathbf{i}\Gamma$ holds, while $a \in \mathbf{o}\Gamma$.

In property 5.6 we present an alternative characterization of "di-initializable".

property 5.6 *di-initializable*

Component Γ is di-initializable if and only if

$$(\mathbf{A}\, a : a \in \mathbf{o}\Gamma : aa \notin \mathbf{t}(\mathbf{dse}\,\Gamma))$$

end of property

In property 5.7 we present a characterization of "di-initializable" in which $\mathbf{dse}\,\Gamma$ does not occur; it shows that in the trace structure that models the communication behavior of di-initializable components all initial 'repetitions' of output symbols are separated by at least one input symbol.

property 5.7 *di-initializable*

Component Γ is di-initializable if and only if

$$(\mathbf{A}\, s, t, a : s \in (\mathbf{o}\Gamma)^* \wedge t \in (\mathbf{o}\Gamma)^* \wedge a \in \mathbf{o}\Gamma : sata \notin \mathbf{t}(\mathbf{ptr}\,\Gamma))$$

end of property

example 5.8

Component Γ_I is defined by:

$$\mathbf{o}\Gamma_I \stackrel{\text{def}}{=} \{a, b\}$$
$$\mathbf{i}\Gamma_I \stackrel{\text{def}}{=} \{c\}$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma_I) \stackrel{\text{def}}{=} \mathbf{pref}\{abca, bc\}$$

Using $abca\mathbf{c}_{\mathbf{io}\Gamma_I}bcaa$, we derive that $\mathbf{t}(\mathbf{dse}\,\Gamma_I) = \mathbf{pref}\{abca, baca, bcaa\}$ from definition 4.36, "**dse**". From definition 5.5, "di-initializable", we conclude that Γ_I is a di-initializable component. Notice that the environment of Γ_I may refuse to send a commsig to which c is associated.

end of example

5.1.0.1 Delay-insensitive enclosure

In this subsection we define for di-initializable component Γ component $\mathbf{DIE}\,\Gamma$, i.e. the delay-insensitive enclosure of Γ. Furthermore, we give a constructive definition of $\mathbf{ptr}(\mathbf{DIE}\,\Gamma)$, viz. $\mathbf{die}\,\Gamma$.

When di-initializable component Γ communicates via a delay-insensitive channel, trace structure $\mathbf{ptr}(\mathbf{DIE}\,\Gamma)$ gives the maximal communication in this channel. Furthermore, component $\overline{\mathbf{DIE}\,\Gamma}$ is the maximal (w.r.t. trace structure inclusion) partner of Γ. When Γ and $\overline{\mathbf{DIE}\,\Gamma}$ are indirectly connected, they have no computation interference hazard and there is no transmission interference hazard in the communication between them.

definition 5.9 *delay-insensitive enclosure*

For di-initializable component Γ, we define the *delay-insensitive enclosure* of Γ, denoted by $\mathbf{DIE}\,\Gamma$, as the maximal (w.r.t. trace structure inclusion) component such that

(i) $\mathbf{io}\,\Gamma = \mathbf{io}(\mathbf{DIE}\,\Gamma)$

(ii) $\Gamma\ \mathit{NCIHDS}\ \overline{\mathbf{DIE}\,\Gamma}$

(iii) $(\mathbf{A}\,a,t : a \in \mathbf{i}(\overline{\mathbf{DIE}\,\Gamma}) \wedge ta \in \mathbf{t}(\mathbf{ptr}(\overline{\mathbf{DIE}\,\Gamma})) : (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,ta))$

(iv) $(\mathbf{ptr}(\overline{\mathbf{DIE}\,\Gamma}), \mathbf{ab}\Gamma) \in \mathbf{C}_4$

end of definition

The existence of the maximum in definition 5.9, "delay-insensitive enclosure", above follows from the "delay-insensitive enclosure" theorem 5.24. Requirement (iii) in definition 5.9 restricts in the traces of $\overline{\mathbf{DIE}\,\Gamma}$ the occurrences of symbols $a \in \mathbf{i}(\overline{\mathbf{DIE}\,\Gamma})$ to those occurrences that are associated with the reception by $\overline{\mathbf{DIE}\,\Gamma}$ of commsigs that may have been sent by Γ. In requirement (iii) there is no need to quantify over symbols $a \in \mathbf{o}(\overline{\mathbf{DIE}\,\Gamma})$, since their occurrences in traces of $\mathbf{ptr}(\overline{\mathbf{DIE}\,\Gamma})$ are restricted by requirement (ii). Compared to definition 4.34, "delay-safe enclosure", we have added requirement (iv). This additional requirement guarantees absence of transmission interference hazard. In order to achieve absence of transmission interference, it is formally sufficient to require only that the second conjunct in definition 5.1, "$\mathbf{C}_4$", holds. However, we are only able to interpret absence of transmission interference hazard by that conjunct within the context of delay-safe communication. For this reason we prefer requirement (iv) in definition 5.9.

For component Γ we will define trace structure $\mathbf{die}\,\Gamma$. It will turn out that this is the trace structure of the delay-insensitive enclosure of Γ, see theorem 5.24, "delay-insensitive enclosure". In order to define $\mathbf{die}\,\Gamma$, we first introduce trace set $\mathbf{tih}\,\Gamma$ and component $\mathbf{DSENTIH}\,\Gamma$. Trace set $\mathbf{tih}\,\Gamma$ will be used to exclude the trace set that is associated with transmission interference hazard from trace structure $\mathbf{dse}\,\Gamma$; in this way component $\mathbf{DSENTIH}\,\Gamma$ is a 'reduction' of component $\mathbf{DSE}\,\Gamma$.

definition 5.10 **tih**

For component Γ, let D be the alphbip that is associated with $\mathbf{io}\,\Gamma$; we define trace set $\mathbf{tih}\,\Gamma$:

$$\mathbf{tih}\,\Gamma \stackrel{\text{def}}{=} \{s,t,a : s \in (\mathbf{a}\Gamma)^* \wedge a \in \mathbf{a}\Gamma \wedge t \in (\mathbf{spa}(a,D))^* \wedge sata \in \mathbf{t}(\mathbf{dse}\,\Gamma) : sata\}$$

end of definition

In definition 5.10, "**tih**", we use a formula that is similar to the condition in property 5.7, "di-initializable".

In definition 5.11 we transform transmission interference hazard into computation interference hazard, see subsection 3.3.0. The operator **dse** in this definition is not present to establish absence of computation interference hazard, but it provides the context in which we address transmission interference hazard.

definition 5.11 **DSENTIH**

For di-initializable component Γ, component $\mathbf{DSENTIH}\,\Gamma$ is defined by:

$$\mathbf{io}(\mathbf{DSENTIH}\,\Gamma) \stackrel{\text{def}}{=} \mathbf{io}\,\Gamma$$

$$\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma) \stackrel{\text{def}}{=} \mathbf{redts}(\mathbf{dse}\,\Gamma, \mathbf{i}\,\Gamma, \mathbf{tih}\,\Gamma)$$

end of definition

In definition 5.11, "**DSENTIH**", the di-initializability of Γ is needed to achieve that $(\mathbf{A}\,s : s \in \mathbf{tih}\,\Gamma : I(s \restriction \mathbf{i}\,\Gamma) > 0)$. Now, we infer from property 1.40 that $\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)$ is nonempty; the prefix-closedness of $\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)$ follows from property 1.40, using the prefix-closedness of $\mathbf{dse}\,\Gamma$.

In property 5.12 we present an alternative characterization of **DSENTIH**.

property 5.12

For di-initializable component Γ,

$$\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma) =$$
$$\mathbf{redts}(\mathbf{dse}\,\Gamma, \mathbf{i}\,\Gamma, \{s,a : s \in (\mathbf{a}\Gamma)^* \wedge a \in \mathbf{a}\Gamma \wedge saa \in \mathbf{t}(\mathbf{dse}\,\Gamma) : saa\})$$

end of property

Our motivation for choosing the noun "**DSENTIH**" is provided by the alternative characterization of **DSENTIH** in property 5.13.

property 5.13

For di-initializable component Γ,

$$\mathbf{io}(\mathbf{DSENTIH}\,\Gamma) = \mathbf{io}(\mathbf{DSE}\,\Gamma)$$

$$\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma) = \mathbf{redts}(\mathbf{ptr}(\mathbf{DSE}\,\Gamma), \mathbf{i}(\mathbf{DSE}\,\Gamma), \mathbf{tih}(\mathbf{DSE}\,\Gamma))$$

end of property

In definition 5.14 the operator **dse** is used to establish absence of computation interference hazard when the communication is delay-safe. Since we have transformed transmission interference hazard into computation interference hazard by definition 5.11, "**DSENTIH**", we also establish absence of transmission interference hazard by doing so.

definition 5.14 **die**

For di-initializable component Γ we define trace structure $\mathbf{die}\,\Gamma$ by

$$\mathbf{die}\,\Gamma \stackrel{\text{def}}{=} \mathbf{dse}(\mathbf{DSENTIH}\,\Gamma)$$

end of definition

The following properties and lemmas are used in the proof of theorem 5.24, "delay-insensitive enclosure".

property 5.15

For di-initializable component Γ,

(i) $\mathbf{die}\,\Gamma$ is nonempty,

(ii) $\mathbf{die}\,\Gamma$ is prefix-closed.

end of property

lemma 5.16

For di-initializable component Γ,

$$(\mathbf{A}\,t, u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\,\Gamma}\,u \wedge u \in \mathbf{t}(\mathbf{die}\,\Gamma) : t \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)))$$

end of lemma

lemma 5.17

For di-initializable component Γ,

$$\mathbf{die}\,\Gamma \subseteq \mathbf{dse}\,\Gamma$$

end of lemma

From lemma 5.17 and lemma 4.43 we infer property 5.18.

property 5.18

For di-initializable component Γ,

$$(\mathbf{A}\, t : t \in \mathbf{t}(\mathbf{die}\,\Gamma) : (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\, t))$$

end of property

lemma 5.19

For di-initializable component Γ,

$$\mathbf{die}\,\Gamma \subseteq \mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)$$

end of lemma

lemma 5.20

For di-initializable component Γ,

$$(\mathbf{die}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{C}_4$$

end of lemma

Absence of computation interference hazard is reflected by the following properties. The condition $\mathbf{io}\,\overline{\Gamma} = \mathbf{io}\,\Delta$ models that Γ and Δ have a closed connection.

property 5.21

For di-initializable components Γ and Δ such that $\mathbf{io}\,\overline{\Gamma} = \mathbf{io}\,\Delta$ and $\mathbf{ptr}\,\Delta = \mathbf{die}\,\Gamma$,

$$(\mathbf{DSENTIH}\,\Gamma)\ \mathit{NCIHDS}\ \Delta$$

end of property

property 5.22

For di-initializable components Γ and Δ such that $\mathbf{io}\,\overline{\Gamma} = \mathbf{io}\,\Delta$ and $\mathbf{ptr}\,\Delta = \mathbf{die}\,\Gamma$,

$$(\mathbf{DSENTIH}\,\Gamma)\ \mathit{NCIHDS}\ \Delta \Rightarrow \Gamma\ \mathit{NCIHDS}\ \Delta$$

end of property

From lemma 4.44 we infer property 5.23.

property 5.23

For di-initializable components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$ and $\overline{\Delta}\ \mathit{NCIHADS}\ (\mathbf{DSENTIH}\,\Gamma)$,

$$(\mathbf{A}\, t, u : t \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)) \wedge u \in (\mathbf{t}(\mathbf{ptr}\,\Delta) \setminus \mathbf{t}(\mathbf{die}\,\Gamma)) : \neg(t\mathbf{c}_{\mathbf{io}\,\Gamma}\, u))$$

end of property

Now, we can link trace structure $\mathbf{die}\,\Gamma$ to component $\mathbf{DIE}\,\Gamma$.

theorem 5.24 *delay-insensitive enclosure*

For di-initializable component Γ,

$$\mathbf{ptr}(\mathbf{DIE}\,\Gamma) = \mathbf{die}\,\Gamma$$

end of theorem

Van der Heijden and Teunissen have developed software to calculate **die** Γ for di-initializable components Γ that have regular trace structures, see [van der Heijden – Teunissen 89]. From their work we infer theorem 5.25, which is presented here without a proof.

theorem 5.25

For di-initializable component Γ,

"**ptr** Γ is regular" $\Rightarrow$ "**die** Γ is regular"

end of theorem

5.1.0.2 Properties of delay-insensitive enclosure

In this subsection we present some properties of **DIE**; of course, see theorem 5.24, "delay-insensitive enclosure", some properties of **tih** and **DSENTIH** are included.

property 5.26

For di-initializable component Γ,

$$(\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{C}_4 \;\Rightarrow\; (\mathbf{tih}\,\Gamma = \varnothing)$$

end of property

Using property 5.26 we infer property 5.27.

property 5.27

For di-initializable component Γ,

$$(\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{C}_4 = (\mathbf{die}\,\Gamma = \mathbf{ptr}\,\Gamma)$$

end of property

We consider a di-initializable component Γ. In order to calculate $\mathbf{die}\,\Gamma$, we first calculate trace structure $\mathbf{dse}\,\Gamma$, see definition 5.11, "**DSENTIH**"; next, we reduce $\mathbf{dse}\,\Gamma$ to $\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)$, cf. definition 1.34, "**redts**"; finally, we calculate trace structure $\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma)$, see definition 5.14, "**die**". Example 5.28 and remark 5.29 show that the approach "first reducing trace structure $\mathbf{ptr}\,\Gamma$ in some way and next calculating **dse** only once" does not work.

example 5.28

Di-initializable component Γ_2 is defined by:

$$\mathbf{o}\Gamma_2 \stackrel{\text{def}}{=} \{a, b\}$$
$$\mathbf{i}\Gamma_2 \stackrel{\text{def}}{=} \{c\}$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma_2) \stackrel{\text{def}}{=} \mathbf{pref}\{abca\}$$

We derive that $\mathbf{t}(\mathbf{dse}\,\Gamma_2) = \mathbf{pref}\{abca, baca, bcaa\}$ from definition 4.36, "**dse**", see also example 5.8. From definition 5.14, "**die**", we derive that $\mathbf{t}(\mathbf{die}\,\Gamma_2) = \mathbf{pref}\{abca, baca\}$. Let Δ_2 be a component such that $\mathbf{io}\,\Delta_2 = \mathbf{io}\,\Gamma_2$, $\mathbf{ptr}\,\Delta_2 \subseteq \mathbf{ptr}\,\Gamma_2$, and $\mathbf{ptr}\,\Delta_2 \neq \mathbf{ptr}\,\Gamma_2$. Since $\mathbf{ptr}\,\Delta_2$ is prefix-closed, $\mathbf{t}(\mathbf{ptr}\,\Delta_2) \subseteq \{\varepsilon, a, ab, abc\}$. Hence, $abca \notin \mathbf{t}(\mathbf{dse}\,\Delta_2)$. As a consequence, $\mathbf{die}\,\Delta_2 \neq \mathbf{die}\,\Gamma_2$.

end of example

remark 5.29

From example 5.28 we conclude that for a component Γ it is, in general, not possible to reduce $\mathbf{ptr}\,\Gamma$ to $\mathbf{ptr}\,\Delta$ for some component Δ such that $\mathbf{io}\,\Delta = \mathbf{io}\,\Gamma$ and $\mathbf{die}\,\Delta = \mathbf{die}\,\Gamma$.

end of remark

From theorem 5.24, "delay-insensitive enclosure", lemma 5.20, and property 5.27 we derive that **DIE** is idempotent.

property 5.30 **DIE** *is idempotent*

For di-initializable component Γ,

$$\mathbf{DIE}(\mathbf{DIE}\,\Gamma) = \mathbf{DIE}\,\Gamma$$

end of property

Like **DSE**, operator **DIE** is not monotonic.

remark 5.31

In general, **DIE** is not monotonic, see example 4.58. For components Γ_5 and Δ_5 in example 4.58, $\mathbf{die}\,\Gamma_5 = \mathbf{dse}\,\Gamma_5$ and $\mathbf{die}\,\Delta_5 = \mathbf{dse}\,\Delta_5$.

end of remark

property 5.32

For i/o-connectable components Γ and Δ such that Γ is di-initializable,

(i) $(\mathbf{DSE}\,\Gamma)\,\mathit{NCIHADS}\,\Delta \Rightarrow (\mathbf{DIE}\,\Gamma)\,\mathit{NCIHADS}\,\Delta$

(ii) $\Delta\,\mathit{NCIHADS}\,(\mathbf{DIE}\,\Gamma) \Rightarrow \Delta\,\mathit{NCIHADS}\,(\mathbf{DSE}\,\Gamma)$

end of property

From lemma 5.16, definition 5.14, "**die**", and lemma 4.49 we infer lemma 5.33.

lemma 5.33

For di-initializable component Γ,

$$(\mathbf{A}\,t, u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\,\Gamma}\,u \wedge u \in \mathbf{t}(\mathbf{die}\,\Gamma) : t \in \mathbf{t}(\mathbf{die}\,\Gamma))$$

end of lemma

From theorem 4.80 we infer theorem 5.34.

theorem 5.34

For di-initializable components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$, $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$, and $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$,

$$\mathbf{die}\,\Gamma = \mathbf{die}\,\Delta$$

end of theorem

We present some examples of **die** and **DIE**.

example 5.35

For component Γ_w, see example 2.47 and example 4.38, $\mathbf{die}\,\Gamma_w = \mathbf{dse}\,\Gamma_w$. Also for Γ_c, see example 2.48, Γ_f, see example 2.49, Γ_{af}, see example 2.49 and example 4.39, Γ_{tubc}, see example 2.50 and example 4.60, Γ_{bc}, see example 2.50 and example 4.61, and Γ_{2w}, see example 2.50, the **die** is equal to the **dse**.

end of example

example 5.36

We consider component Γ_{or}, see example 2.51. The state graph of **DIE** Γ_{or} is shown in figure 5.0.

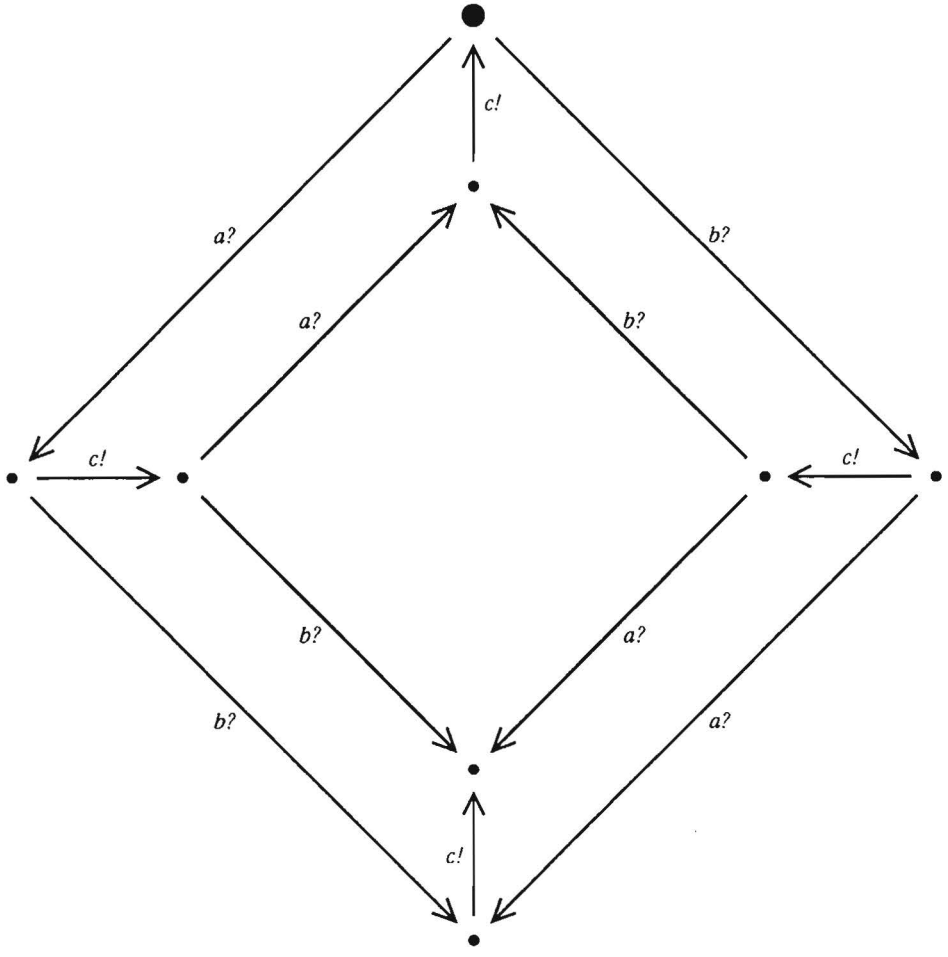

figure 5.0
State graph of component **DIE** Γ_{or}.

Notice that

$\mathbf{t}(\mathbf{die}\,\Gamma_{or}) = \mathbf{pref}\,\{x, y : x \in \{acac, bcbc\}^* \wedge y \in \{acb, abc, bca, bac\} : xy\}$.

There are some traces in $\mathbf{t}(\mathbf{die}\,\Gamma_{or})$ that 'lead to dead ends', viz. all traces in $\{x,y : x \in \{acac, bcbc\}^* \wedge y \in \mathbf{pref}\,\{acb, abc, bca, bac\} : xy\}$. We consider such a trace which is apparently not extendable: *acb*.

(i) Since $(\mathbf{A}\,s : s \in \mathbf{ptr}\,\Gamma_{or} : \neg(s\mathbf{c}_{\mathbf{io}\,\Gamma_{or}}\,acbc))$, we conclude from property 5.18 that $acbc \notin \mathbf{t}(\mathbf{die}\,\Gamma_{or})$.

(ii) From lemma 5.20 and theorem 5.2, "C_4", we conclude that $acbb \notin \mathbf{t}(\mathbf{die}\,\Gamma_{or})$.

(iii) From lemma 5.20, definition 5.1, "C_4", definition 4.16, "D_4", and theorem 5.2, "C_4", we conclude, using $acacbc \in \mathbf{t}(\mathbf{die}\,\Gamma_{or})$, $acacbc\,\mathbf{c}_{\mathbf{io}\,\Gamma_{or}}\,acbacc$, and $acbacc\,\mathbf{c}_{\mathbf{io}\,\Gamma_{or}}\,acba$, that $acba \notin \mathbf{t}(\mathbf{die}\,\Gamma_{or})$.

The non-extendability of the other traces 'leading to dead ends' can be argued analogously. The interpretation of the existence of such traces is the following: when some environment communicates delay-insensitively with component Γ_{or} in such a way as to 'move $\mathbf{DIE}\,\Gamma_{or}$ to a dead end', component Γ_{or} goes along without violating any of the correctness concerns, viz. absence of computation interference hazard and absence of transmission interference hazard; however, any further extension of the communication will violate at least one of the correctness concerns. In this particular example the correctness concern "absence of transmission interference hazard" will be violated.

end of example

example 5.37

We consider component Γ_{and}, see example 2.51. The state graph of **DIE** Γ_{and} is shown in figure 5.1.

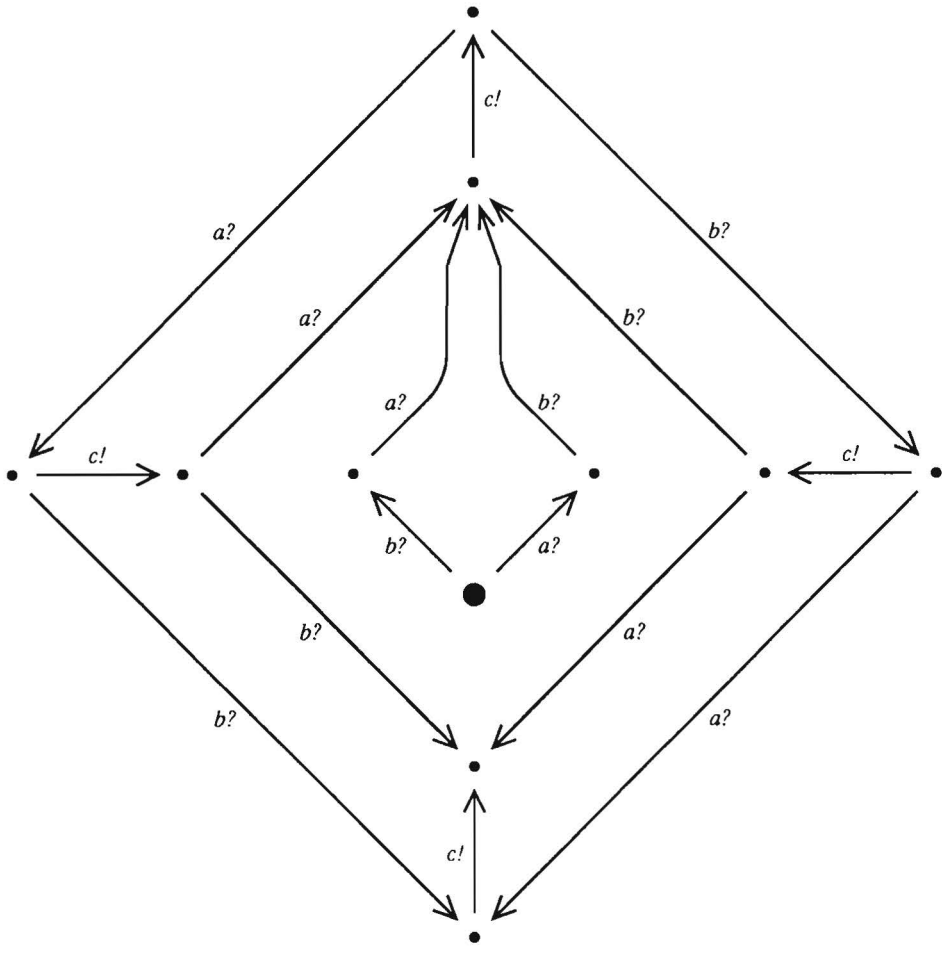

figure 5.1
State graph of component **DIE** Γ_{and}.

Notice that **DIE** Γ_{and} differs from **DIE** Γ_{or}, see example 5.36, only in its initial behavior, cf. example 2.51.

end of example

Component Γ_{ncel}, see example 5.38, has been presented by Ebergen, see [Ebergen 87].

example 5.38

We consider Ebergen's "NCEL component", cf. [Ebergen87]. We call it Γ_{ncel}; it is defined by:

$$\mathbf{o}\Gamma_{ncel} \stackrel{\text{def}}{=} \{c\} \qquad \mathbf{i}\Gamma_{ncel} \stackrel{\text{def}}{=} \{a, b\}$$

$$\mathbf{t}(\mathbf{ptr}\,\Gamma_{ncel}) \stackrel{\text{def}}{=} \mathbf{pref}(\{aa, bb, abc, bac\}^*),$$

We present the state graph of component Γ_{ncel} in figure 5.2.

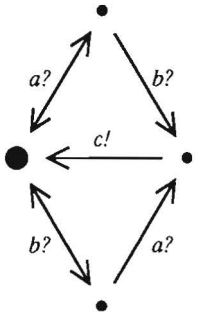

figure 5.2
State graph of component Γ_{ncel}.

From theorem 5.24, "delay-insensitive enclosure", and definition 5.14, "**die**", we conclude that $\mathbf{DIE}\,\Gamma_{ncel} = \Gamma_c$, cf. example 2.48. Ebergen presents Γ_{ncel} as an example of a component that is not a "DI component"; in our terminology this means that $\mathbf{DIE}\,\Gamma_{ncel} \neq \Gamma_{ncel}$.

end of example

5.1.0.3 Behavior of delay-insensitively communicating components

In this subsection we are interested in the impact of delay-insensitive communication on the communication behavior of a component. We define the *maximal communication behavior of a component that communicates delay-insensitively*, i.e. the maximal (w.r.t. trace structure inclusion) communication behavior of the component at the commports of the component when the component has an indirect connection with its environment and there is absence of computation interference hazard between them and there is absence of transmission interference hazard in the channel between them. The "maximal communication behavior of component Γ that communicates delay-insensitively" is a component. It is denoted by $\mathbf{CBDI}\,\Gamma$.

definition 5.39 *maximal communication behavior for delay-insensitive communication*
For di-initializable component Γ, we define the *maximal communication behavior of* Γ *when* Γ *communicates via a delay-insensitive channel*, denoted by $\mathbf{CBDI}\,\Gamma$, as the maximal (w.r.t. trace structure inclusion) component such that

(i) $\mathbf{io}(\mathbf{CBDI}\,\Gamma) = \mathbf{io}\,\Gamma$

(ii) $\mathbf{ptr}(\mathbf{CBDI}\,\Gamma) \subseteq \mathbf{ptr}\,\Gamma$

(iii) $(\mathbf{A}\,a, s : a \in \mathbf{i}\Gamma \wedge sa \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDI}\,\Gamma)) : (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{die}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\,\Gamma}t))$

end of definition

The existence of the maximum in definition 5.39, "maximal communication behavior for delay-insensitive communication", above follows from theorem theorem 5.48, "maximal communication behavior for delay-insensitive communication". In requirement (iii) in definition 5.39, "maximal communication behavior for delay-insensitive communication", we do not quantify over symbols $a \in \mathbf{o}\,\Gamma$, since there is no way to prevent a component to 'produce' commsigs at its output commports, cf. subsection 2.2.3.

property 5.40
For di-initializable component Γ,

$$(\mathbf{CBDI}\,\Gamma)\ \mathit{NCIHDS}\ \overline{\mathbf{DIE}\,\Gamma}$$

end of property

From property 5.40 we conclude that there is no need to require that $(\mathbf{CBDI}\,\Gamma)\ \mathit{NCIHDS}\ \overline{\mathbf{DIE}\,\Gamma}$ in definition 5.39, "maximal communication behavior for delay-insensitive communication".

For component Γ we define trace structure $\mathbf{cbdi}\,\Gamma$. It will turn out that this is the trace structure of $\mathbf{CBDI}\,\Gamma$, see theorem 5.48.

definition 5.41 **cbdi**
For di-initializable component Γ trace structure $\mathbf{cbdi}\,\Gamma$ is defined by

$$\mathbf{cbdi}\,\Gamma \stackrel{\text{def}}{=} < \mathbf{a}\Gamma, \{t, u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\,\Gamma}u \wedge u \in \mathbf{t}(\mathbf{die}\,\Gamma) : t\} >$$

end of definition

property 5.42
For di-initializable component Γ, trace t, and symbol a,

(i) for $a \in \mathbf{o}\,\Gamma$, $\quad (t \in \mathbf{t}(\mathbf{cbdi}\,\Gamma) \wedge ta \in \mathbf{t}(\mathbf{ptr}\,\Gamma)) = ta \in \mathbf{t}(\mathbf{cbdi}\,\Gamma)$

(ii) for $a \in \mathbf{i}\,\Gamma$, $\quad (t \in \mathbf{t}(\mathbf{cbdi}\,\Gamma) \wedge ta \in \mathbf{t}(\mathbf{die}\,\Gamma)) = ta \in \mathbf{t}(\mathbf{cbdi}\,\Gamma)$

end of property

In property 5.42 (ii) $ta \in \mathbf{t}(\mathbf{cbdi}\,\Gamma) \Rightarrow ta \in \mathbf{t}(\mathbf{die}\,\Gamma)$ follows from lemma 5.33; from lemma 5.33 we also infer property 5.43.

property 5.43

For di-initializable component Γ,

$$\mathbf{cbdi}\,\Gamma = \mathbf{ptr}\,\Gamma \cap \mathbf{die}\,\Gamma$$

end of property

theorem 5.44

For di-initializable component Γ,

$$\mathbf{cbdi}\,\Gamma = \mathbf{cbds}\,\Gamma \cap \mathbf{die}\,\Gamma$$

end of theorem

remark 5.45

In example 4.60 we have seen that the correctness concern "absence of computation interference hazard" restricts the communication behavior of a component. From theorem 5.44 we conclude that the additional correctness concern "absence of transmission interference hazard" indeed gives an additional restriction on the communication behavior of a component, cf. example 5.46.

end of remark

example 5.46

Component Γ_3 is defined by:

$$\mathbf{o}\,\Gamma_3 \stackrel{\text{def}}{=} \varnothing$$
$$\mathbf{i}\,\Gamma_3 \stackrel{\text{def}}{=} \{a, b\}$$
$$\mathbf{t}(\mathbf{ptr}\,\Gamma_3) \stackrel{\text{def}}{=} \{\varepsilon, a, b, ba, aa\}$$

We infer that $\mathbf{t}(\mathbf{cbds}\,\Gamma_3) = \{\varepsilon, a, b, aa\}$ and $\mathbf{t}(\mathbf{cbdi}\,\Gamma_3) = \{\varepsilon, a, b\}$. Since $\mathbf{cbds}\,\Gamma_3 = \mathbf{die}\,\Gamma_3$, we conclude that the communication behavior of Γ_3 is further restricted by the additional correctness concern absence of transmission interference hazard.

end of example

Due to the nonemptiness and prefix-closedness of **ptr** and **dse** we infer property 5.47 from property 5.43.

property 5.47

For di-initializable component Γ,

$\mathbf{cbdi}\,\Gamma$ is nonempty and prefix-closed

end of property

Now, we can link trace structure $\mathbf{cbdi}\,\Gamma$ to component $\mathbf{CBDI}\,\Gamma$.

theorem 5.48 *maximal communication behavior for delay-insensitive communication*
For di-initializable component Γ,

$$\mathbf{ptr}(\mathbf{CBDI}\,\Gamma) = \mathbf{cbdi}\,\Gamma$$

end of theorem

Property 5.49 relates the trace structures of components $\mathbf{DSENTIH}\,\Gamma$ and $\mathbf{CBDI}\,\Gamma$.

property 5.49
For di-initializable components Γ and Δ such that $\mathbf{io}\,\overline{\Gamma} = \mathbf{io}\,\Delta$,

$$(\mathbf{DSENTIH}\,\Gamma)\ \mathit{NCIHDS}\ \Delta$$
$$\Rightarrow\ (\mathbf{A}\,t,u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\,\Gamma}\,u \wedge u \in \mathbf{t}(\mathbf{ptr}\,\Delta)$$
$$: t \in \mathbf{t}(\mathbf{cbdi}\,\Gamma) \wedge u \in \mathbf{t}(\mathbf{cbdi}\,\Delta)$$
$$)$$

end of property

From theorem 5.34 and definition 5.39, "maximal communication behavior for delay-insensitive communication" we derive that **CBDI** is idempotent.

property 5.50 **CBDI** *is idempotent*
For di-initializable component Γ,

$$\mathbf{CBDI}(\mathbf{CBDI}\,\Gamma) = \mathbf{CBDI}\,\Gamma$$

end of property

In example 5.51 we take another look at Ebergen's Γ_{ncel}.

example 5.51
We consider Ebergen's "NCEL component", cf. [Ebergen 87] and example 5.38.

Using property 5.43 we calculate trace structure $\mathbf{cbdi}\,\Gamma_{ncel}$. From theorem 5.48, "maximal communication behavior for delay-insensitive communication", we now conclude that $\mathbf{CBDI}\,\Gamma_{ncel} = \Gamma_c$, cf. example 2.48.

end of example

5.1.1 'Off-the-shelf' mechanisms

After we have determined the trace structure of the component that models the mechanism, we calculate the **cbdi** of this component. If this **cbdi** is too restricted, we conclude that we do not want to require that the communication between this component and its environment is delay-insensitive. We might choose a mixed connection between this component and its environment, cf. partial delay-insensitivity in chapter 6; or we might create a new mechanism, e.g. by adding some clock signals as described in subsection 4.2.5.

In subsection 5.1.0.1 we have defined the operator **die** for di-initializable components. For di-initializable component Γ, **die**Γ is the trace structure of the delay-insensitive enclosure of Γ. Using **die** we have defined the operator **cbdi** in subsection 5.1.0.3; **cbdi**Γ is the trace structure of the maximal communication behavior of di-initializable component Γ, if Γ communicates delay-insensitively. Let di-initializable component Γ model some 'off-the-shelf' mechanism; now, trace structure **cbdi**Γ can be used as a label that can be attached to such a mechanism to show its maximal delay-insensitive communication behavior. This labeling can be done only for di-initializable components; notice that labeling such an 'off-the-shelf' mechanism with trace structure **cbds**Γ to show its maximal delay-safe communication behavior is always possible, see subsection 4.2.5. For components that are not di-initializable, the **cbdi** is not defined: such a component cannot be prevented from causing transmission interference hazard when it communicates via a delay-safe channel. This problem might be solved by assuming that the connection between this component and its environment is mixed, cf. partial delay-insensitivity in chapter 6.

6

Composition

In this chapter we present our most general study of composition: we are concerned with mixed connections of components, cf. subsection 2.1.3.0 and subsection 2.2.3. In section 6.0 we introduce some notions that are used in the following sections. In section 6.1 we study composition of components given the correctness concern "absence of computation interference hazard". We deal with an additional correctness concern, viz. "absence of transmission interference hazard", in section 6.2. We address decomposition in section 6.3. We refer to other correctness concerns in section 6.4.

6.0 Connection of components

In this chapter we study the composition of two components; in order to refer to them conveniently in the remainder of this chapter, we call them Γ and Δ. There are several ways in which two components can be connected. In subsection 2.2.0 we have stated that we associate the same symbol with two matching commports. In order to compose two components they have to be i/o-connectable, cf. definition 3.2.

In chapters 3, 4, and 5 we studied closed connections of two components, cf. subsection 2.1.3.0. In chapter 6 we do not restrict ourselves to closed connections: we are interested in connections of two components that are either open or closed. Furthermore, in chapter 3 we have dealt with direct connections of two components; in chapters 4 and 5 we have dealt with indirect connections. The clearly missing case of *mixed connections* will emerge from the treatment of composition in chapter 6; within a mixed connection of components, some pairs of matching commports may be directly connected, whereas the others may be indirectly connected.

remark 6.0

The open composition of two components constitutes a problem in which implicitly the environment of these components appears as a third component. As a consequence, we must deal with the closed composition of three components. By calculating the composite of two components given some correctness concerns, we generate conditions on the acceptance of (external) inputs by the composite. If absence of computation interference hazard is a correctness concern for the communication between such a composite and its environment, the allowed communication behavior of this environment is reduced by these conditions on the acceptance of (external) inputs by the composite.

end of remark

Some people do not care about the environment. We do care about the environment, since we are interested in the correctness concern "absence of computation interference hazard" for the communication between the composite and its environment. As a consequence, when discussing the open composition of two components we address three party composition. We do not explicitly refer to the environment of the composite. Nevertheless, concerns about this environment are present: we address this environment implicitly.

remark 6.1

When calculating the composite, we assume that the (implicit) environment of this composite is directly connected to this composite.

end of remark

6.0.0 External input and output

Since components that have an open connection participate in communication with the environment of their composite, we define the notions "*external input*" and "*external output*".

definition 6.2 **extinp**

For i/o-connectable components Γ and Δ, alphabet **extinp**(Γ, Δ) is defined by

$$\textbf{extinp}(\Gamma, \Delta) \stackrel{\text{def}}{=} (\mathbf{i}\Gamma \cup \mathbf{i}\Delta) \cap (\mathbf{a}\Gamma \div \mathbf{a}\Delta)$$

end of definition

definition 6.3 **extoutp**

For i/o-connectable components Γ and Δ, alphabet **extoutp**(Γ, Δ) is defined by

$$\textbf{extoutp}(\Gamma, \Delta) \stackrel{\text{def}}{=} (\mathbf{o}\Gamma \cup \mathbf{o}\Delta) \cap (\mathbf{a}\Gamma \div \mathbf{a}\Delta)$$

end of definition

The set **extinp**(Γ, Δ) is associated with the set of *external input commports* of i/o-connectable components Γ and Δ. The set **extoutp**(Γ, Δ) is associated with the set of *external output commports* of i/o-connectable components Γ and Δ. In our Communication Model all communication is one-to-one communication, cf. subsection 2.1.0. As a consequence, inputs and outputs of Γ and Δ that belong to $\mathbf{a}\Gamma \cap \mathbf{a}\Delta$ are not available for external communication; for this reason these inputs and outputs are excluded from **extinp**(Γ, Δ) and **extoutp**(Γ, Δ). In definition 6.2, "**extinp**", and definition 6.3, "**extoutp**", the i/o-connectability of Γ and Δ is required to provide the context for these definitions.

When we study composition of components in our Communication Model, we distinguish directly and indirectly connected commports, cf. subsection 2.1.0. Since in this chapter we are concerned with mixed connections of components, we deal with both cases. In the definitions in this chapter, alphabets I and D are used to indicate which part of the connection (of components Γ and Δ) is indirect and which part is direct: the symbols in $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \cap I$ are associated with the *indirectly connected commports* of these components, and the symbols in $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \cap D$ are associated with their *directly connected commports*, see figure 6.0. Set I is associated with the set Ψ_{ic} of indirectly connected commports, see subsection 2.1.6.0; as a consequence, set D is associated with $\Psi \setminus \Psi_{ic}$.

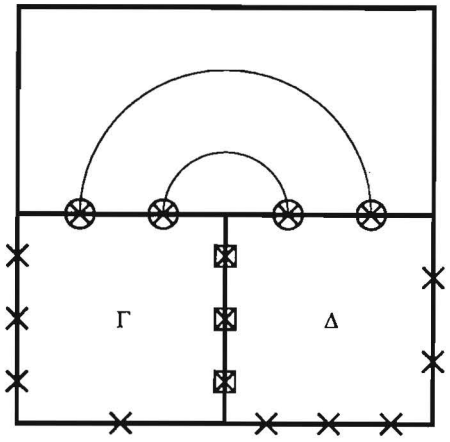

figure 6.0
Composing components Γ and Δ under I.

In figure 6.0 we distinguish three kinds of commports of the composite of components Γ and Δ:

- the external commports, with which an element of $\mathbf{a}\Gamma \div \mathbf{a}\Delta$ is associated.
- the indirectly connected matching commports of Γ and Δ, with which an element of $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \cap I$ is associated; these commports have been encircled.
- the directly connected matching commports of Γ and Δ, with which an element of $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \cap D$ is associated; these commports have been boxed.

Alphabets I and D constitute a bipartition of the universe Ω of symbols: $D = \Omega \backslash I$. In the remainder of this monograph we will use the expression *the composition of components under I* as an abbreviation for "the composition of components in which with the indirectly connected matching commports an element of I is associated and in which with the directly connected matching commports an element of D is associated".

remark 6.4

Many definitions in this chapter depend on the bipartition of Ω into alphabets I and D. As a consequence, I occurs formally as a parameter in these definitions. For this reason, we explicitly mention I in these definitions rather than defining it globally with respect to them.

end of remark

6.0.1 General composability

When two components have a mixed connection, we say that the communication between them is *partially delay-safe*, cf. [Schols86]. In order to deal with the (partial) delay-safe communication in a composition of two components we extend definition 4.1, "composability", leading to definition 6.5, "general composability". In definition 6.5, (i) through (v) are equal to (i) through (v) in definition 4.1. Condition (vi) is added in definition 6.5. We do not use *general composability* to relate traces of Γ to traces of Δ; we use it to give a relation between traces (at the curly boundary in figure 6.1, see also remark 6.1) of the composite of Γ and Δ. We project these traces (onto $\mathbf{a}\Gamma$ or $\mathbf{a}\Delta$) when we want to relate them to traces of either Γ or Δ, see also definition 6.11, "*totcom*".

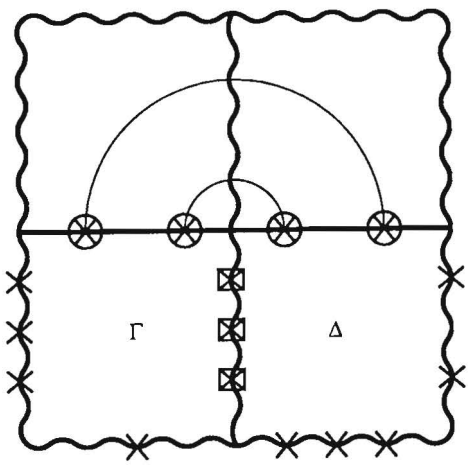

figure 6.1
Composing components Γ and Δ.

Definition 6.5, "general composability", will be used such that iobip F is equal to the restriction of iobip $\mathbf{io}\Gamma$ to the indirect connection of Γ and Δ: $\mathbf{i}F=(\mathbf{i}\Gamma\cap\mathbf{a}\Delta\cap I)$ and $\mathbf{o}F=(\mathbf{o}\Gamma\cap\mathbf{a}\Delta\cap I)$. This relation between Γ, Δ, and F is shown in definition 6.11, "*totcom*". As a consequence, each symbol $c\notin\mathbf{a}F$ in definition 6.5(vi) is associated with either a pair of directly connected commports of Γ and Δ or a commport of Γ or Δ that does not match a commport of the other component: $(\mathbf{a}\Gamma\cap\mathbf{a}\Delta)\backslash\mathbf{a}F=(\mathbf{a}\Gamma\cap\mathbf{a}\Delta\cap D)\cup(\mathbf{a}\Gamma\div\mathbf{a}\Delta)$. Thus, $\mathbf{a}F$ includes only symbols that are associated with the encircled commports of figure 6.1. The boxed commports of figure 6.1 are directly connected.

We use *general composability* to capture our causality notion, cf. section 4.0. Since in definition 6.5, "general composability" (ii) through (v) we are concerned with symbols that are associated with indirectly connected commports, the consistency with our causality notion follows from section 4.0. For this reason (ii) through (v) in definition 6.5, are equal to (ii) through (v) in definition 4.1, "composability". A symbol that is associated with a pair of directly connected commports is added to both general composable traces, see definition 6.5(vi), in order to maintain consistency with our causality notion, see also chapter 3 and section 4.0. A symbol that is associated with a commport of Γ or Δ that does not match a commport of the other component is also added to both general composable traces, see definition 6.5(vi); since the environment of the composite of Γ and Δ is assumed to be directly connected to this composite, cf. remark 6.1, adding these symbols to both general composable traces maintains consistency with our causality notion.

definition 6.5 *general composability*

For traces t and u and iobip F, we define that t is *generally composable under* F with u, denoted by $t\mathbf{g}_F u$, recursively by

(i) $\varepsilon \mathbf{g}_F \varepsilon$

(ii) for traces t and u and symbol a such that $t\mathbf{g}_F u$ and $a \in \mathbf{o}F$,

$$ta\mathbf{g}_F u$$

(iii) for traces t and u and symbol a such that $t\mathbf{g}_F u$, $a \in \mathbf{o}F$, and $\#_a t > \#_a u$,

$$t\mathbf{g}_F ua$$

(iv) for traces t and u and symbol b such that $t\mathbf{g}_F u$ and $b \in \mathbf{i}F$,

$$t\mathbf{g}_F ub$$

(v) for traces t and u and symbol b such that $t\mathbf{g}_F u$, $b \in \mathbf{i}F$, and $\#_b u > \#_b t$,

$$tb\mathbf{g}_F u$$

(vi) for traces t and u and symbol c such that $t\mathbf{g}_F u$ and $c \notin \mathbf{a}F$,

$$tc\mathbf{g}_F uc$$

(vii) completeness axiom: t is not generally composable under F with u, unless this is required by (i), (ii), (iii), (iv), (v), or (vi).

end of definition

From definition 6.5, "general composability", we conclude that, as far as general composability (under iobip F) is concerned, it is irrelevant whether a symbol that is not in $\mathbf{a}F$ is an input or an output with respect to some iobip (see also subsection 6.0.1.0). Of course, when we take absence of computation interference hazard into account, this difference is crucial.

In analogy to property 4.9 we infer property 6.6.

property 6.6

For traces t and u, and iobip F,

$$t\mathbf{g}_F u = u\mathbf{g}_{\bar{F}} t$$

end of property

6.0.1.0 Relation to composability

The relation between definition 6.5, "general composability", and definition 4.1, "composability", is expressed in property 6.7.

property 6.7

For traces t and u and iobips F and F' such that $t \in (\mathbf{a}F)^*$, $u \in (\mathbf{a}F)^*$, $\mathbf{o}F' \subseteq \mathbf{o}F$, and $\mathbf{i}F' \subseteq \mathbf{i}F$,

$$t\mathbf{g}_{F'} u = t\mathbf{c}_F u \wedge (t \restriction (\mathbf{a}F \setminus \mathbf{a}F') = u \restriction (\mathbf{a}F \setminus \mathbf{a}F'))$$

end of property

We consider two i/o-connectable components Γ and Δ. When we want to apply property 6.7, we may want to choose iobips F and F' as follows:

$$\mathbf{o}F' = \mathbf{o}\Gamma \cap \mathbf{a}\Delta \cap I$$

$$\mathbf{i}F' = \mathbf{i}\Gamma \cap \mathbf{a}\Delta \cap I$$

$$\mathbf{o}F = \mathbf{o}\Gamma \cup \mathbf{extoutp}(\Gamma, \Delta)$$

$$\mathbf{i}F = \mathbf{i}\Gamma \cup \mathbf{extinp}(\Gamma, \Delta)$$

The distribution of the external outputs and external inputs over $\mathbf{o}F$ and $\mathbf{i}F$ is irrelevant from a formal point of view.

We consider traces t and u and iobips F and F' such that $t \in (\mathbf{a}F)^*$, $u \in (\mathbf{a}F)^*$, $\mathbf{o}F' \subseteq \mathbf{o}F$, $\mathbf{i}F' \subseteq \mathbf{i}F$, and $t\mathbf{g}_{F'} u$. From property 6.7 we infer that our causality constraint, viz. $t\mathbf{c}_F u$, holds independently of whether symbols are associated with directly or indirectly connected commports; for symbols that are associated with directly connected commports there is an additional constraint, viz. that they have to occur in the same order in both traces.

remark 6.8

In chapter 4 we introduced our causality notion: no commsig is received before it has been sent. This causality notion holds independent of whether a commsig is sent between directly or indirectly connected commports. For directly connected commports there is an additional constraint: sending and reception of a commsig coincide.

end of remark

6.0.1.1 General composability diagram

In subsection 4.0.0 we explained the construction of composability diagrams. This construction method of Verhoeff is used to check whether two traces are composable under an iobip. In this subsection 6.0.1.1 we extend this method. The extended method is called *constructing a general composability diagram*. By constructing a general composability diagram we check whether two traces are generally composable under an iobip.

When constructing a composability diagram, see subsection 4.0.0, we are interested in whether traces t and u are composable under iobip F; here, t, u, and F satisfy $t \in (\mathbf{a}F)^*$ and $u \in (\mathbf{a}F)^*$. We have seen that definition 6.5, "general composability", differs from definition 4.1, "composability", by the addition of (vi). In 6.5(vi) we deal with symbols that are not elements of $\mathbf{a}F$. The construction of a general composability diagram is equal to the construction of a composability diagram with one exception: an occurrence of a symbol that is not an element of $\mathbf{a}F$ is connected by a bidirectional arrow (in stead of a unidirectional arrow). The bidirectional arrow is treated as two non-intersecting arrows that point in opposite directions. A symbol that is not in $\mathbf{a}F$ is associated with two commports that are directly connected, cf. remark 6.1. As a consequence, such a symbol has to occur consitent with our causality notion either in both composable traces or in none, cf. definition 6.5(vi). These symbols which are not in $\mathbf{a}F$ are not postfixed (nor with an exclamation mark nor with a question mark) in the traces in a general composability diagram.

Trace t is generally composable under iobip F with trace u if and only if in the general composability diagram:

(i) there is no arrow starting from a \$, and

(ii) there is no backward intersection of two arrows.

These two conditions are equal to the conditions in subsection 4.0.0.

example 6.9 *generally composable traces*

We consider traces t and u, symbols a, b, c, and d, and iobip F_3 such that $\mathbf{o}F_3 = \{a\}$ and $\mathbf{i}F_3 = \{b, d\}$. We are interested in whether trace $abca$ $(=t)$ is generally composable under F_3 with trace $adbcb$ $(=u)$. In figure 6.2 this general composability diagram is shown.

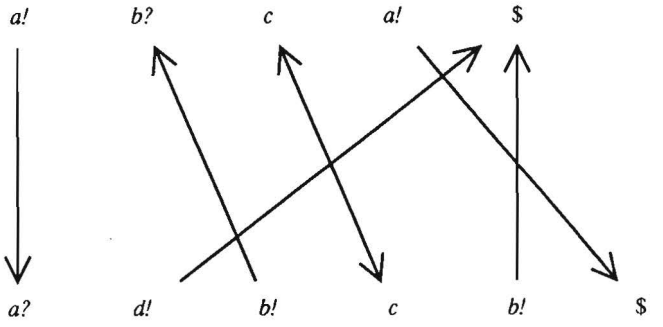

figure 6.2
General composability diagram.

The absence of a backward intersection in the general composability diagram indicates that t and u are composable under F_3. By direct application of definition 6.5, "general composability", we can derive in several ways a confirmation that $abca\mathbf{g}_{F_3}adbcb$:

$\varepsilon \mathbf{g}_{F_3} \varepsilon$	$\varepsilon \mathbf{g}_{F_3} \varepsilon$
$a \mathbf{g}_{F_3} \varepsilon$	$a \mathbf{g}_{F_3} \varepsilon$
$a \mathbf{g}_{F_3} a$	$a \mathbf{g}_{F_3} a$
$a \mathbf{g}_{F_3} ad$	$a \mathbf{g}_{F_3} ad$
$a \mathbf{g}_{F_3} adb$	$a \mathbf{g}_{F_3} adb$
$ab \mathbf{g}_{F_3} adb$	$ab \mathbf{g}_{F_3} adb$
$abc \mathbf{g}_{F_3} adbc$	$abc \mathbf{g}_{F_3} adbc$
$abc \mathbf{g}_{F_3} adbcb$	$abca \mathbf{g}_{F_3} adbc$
$abca \mathbf{g}_{F_3} adbcb$	$abca \mathbf{g}_{F_3} adbcb$

table 6.3
Two derivations of $abca\mathbf{g}_{F_3}adbcb$.

end of example

In example 6.10 we show how the bidirectional arrows are used to check for general composabability.

example 6.10

We consider symbols a and b. We are interested in whether trace ab is generally composable under a given iobip with trace ba. In this example we consider several iobips.

Let iobip F_4 be such that $\mathbf{a}F_4 = \varnothing$. We are interested in whether $ab\mathbf{g}_{F_4}ba$. In figure 6.4 this general composability diagram is shown.

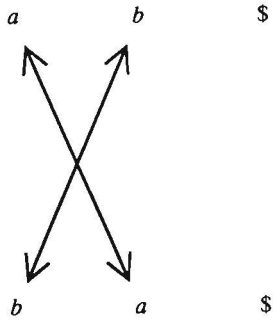

figure 6.4
General composability diagram.

Since the bidirectional arrows give rise to a backward intersection in the general composability diagram, we conclude that $\neg(ab\mathbf{g}_{F_4}ba)$.

Let iobip F_5 be such that $\mathbf{o}F_5 = \{a\}$ and $\mathbf{i}F_5 = \varnothing$. We are interested in whether $ab\mathbf{g}_{F_5}ba$. In figure 6.5 this general composability diagram is shown.

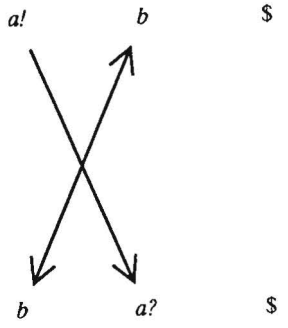

figure 6.5
General composability diagram of $ab\mathbf{g}_{F_5}ba$.

Since there is no a backward intersection in the general composability diagram and there is no arrow starting from a \$, we conclude that $ab\mathbf{g}_{F_5}ba$.

Let iobip F_6 be such that $\mathbf{o}F_6 = \{b\}$ and $\mathbf{i}F_6 = \varnothing$. We are interested in whether $ab\mathbf{g}_{F_6}ba$. In figure 6.6 this general composability diagram is shown.

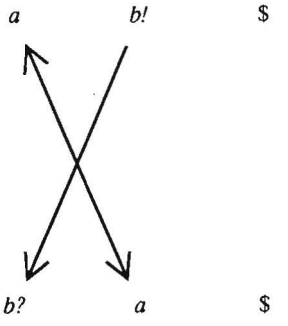

figure 6.6
General composability diagram.

Since the bidirectional arrow gives rise to a backward intersection in the general composability diagram, we conclude that $\neg(ab\mathbf{g}_{F_6}ba)$.

end of example

6.1 Composition without computation interference hazard

When we compose components we define a component that is the composite of them: this composite is under the given correctness concerns maximal with respect to both the (external) inputs that are guaranteed to be accepted by it and the (external) outputs that might be produced by it. In this section we are concerned with only one correctness concern, viz. "absence of computation interference hazard". The composite of two components, say Γ and Δ, given this correctness concern is calculated in three steps:

- We calculate in subsection 6.1.0 trace structure Γ ***totcom***$_I\Delta$, which results from 'combining' the trace structures of Γ and Δ. This combination is calculated regardless of our correctness concern "absence of computation interference hazard", see definition 6.11, "***totcom***". Of course, the alphabet of trace structure Γ ***totcom***$_I\Delta$ equals $a\Gamma \cup a\Delta$.
- In subsection 6.1.1 we deal with the correctness concern "absence of computation interference hazard". Using this correctness concern we calculate trace structure Γ ***totcomncih***$_I\Delta$, see definition 6.24. Of course, the alphabet of trace structure Γ ***totcomncih***$_I\Delta$ equals $a\Gamma \cup a\Delta$. The trace set of Γ ***totcomncih***$_I\Delta$ is a subset of the trace set of Γ ***totcom***$_I\Delta$.
- In subsection 6.1.2 we hide the internal communication; the resulting trace structure is called Γ ***extcomncih***$_I\Delta$. Of course, the alphabet of trace structure Γ ***extcomncih***$_I\Delta$ equals $a\Gamma \div a\Delta$, see definition 6.29, "***extcomncih***". In this step we maintain absence of computation interference hazard, which has been established in the previous step.

Of course, we will motivate the calculations performed in the three steps mentioned above. However, since in our Communication Model we cannot give an interpretation for the trace stuctures calculated in the first two steps, we do not give such an interpretation in our Communication Model: no component is defined in these first two steps. Notice that this implies that we do not attempt to define a kind of 'composite regardless of "absence of computation interference hazard"' in our Communication Model.

6.1.0 Combining two connected components

We combine the trace structures of Γ and Δ into one trace structure, viz. $\Gamma\ totcom_I\ \Delta$, see definition 6.11.

definition 6.11 *totcom*

Given are i/o-connectable components Γ and Δ and alphabet I. Let iobip F be such that $\mathbf{i}F = \mathbf{i}\Gamma \cap \mathbf{a}\Delta \cap I$ and $\mathbf{o}F = \mathbf{o}\Gamma \cap \mathbf{a}\Delta \cap I$. Trace structure $\Gamma\ totcom_I\ \Delta$ is defined by:

(i) $\mathbf{a}(\Gamma\ totcom_I\ \Delta) \stackrel{\text{def}}{=} \mathbf{a}\Gamma \cup \mathbf{a}\Delta$

(ii) $\varepsilon \in \mathbf{t}(\Gamma\ totcom_I\ \Delta)$

(iii) for traces s and t and symbol d such that $d \in (\mathbf{a}\Gamma \setminus \mathbf{o}\Delta)$, $s \in (\mathbf{a}\Gamma \cup \mathbf{a}\Delta)^*$, $t \in \mathbf{t}(\Gamma\ totcom_I\ \Delta)$, $(s \restriction \mathbf{a}\Gamma) \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, and $s\,\mathbf{g}_F\,td$,

$$td \in \mathbf{t}(\Gamma\ totcom_I\ \Delta)$$

(iv) for traces t and u and symbol e such that $e \in (\mathbf{a}\Delta \setminus \mathbf{o}\Gamma)$, $t \in \mathbf{t}(\Gamma\ totcom_I\ \Delta)$, $u \in (\mathbf{a}\Gamma \cup \mathbf{a}\Delta)^*$, $(u \restriction \mathbf{a}\Delta) \in \mathbf{t}(\mathbf{ptr}\,\Delta)$, and $te\,\mathbf{g}_F\,u$,

$$te \in \mathbf{t}(\Gamma\ totcom_I\ \Delta)$$

(v) completeness axiom: $\mathbf{t}(\Gamma\ totcom_I\ \Delta)$ contains no traces that are not required by (ii), (iii), or (iv).

end of definition

In definition 6.11(iii) symbol d is associated with either an external commport of Γ or an internal output commport of Γ, since $(\mathbf{a}\Gamma \setminus \mathbf{o}\Delta) = (\mathbf{a}\Gamma \setminus \mathbf{a}\Delta) \cup (\mathbf{o}\Gamma \cap \mathbf{i}\Delta)$. Analogously, in definition 6.11(iv) symbol e is associated with either an external commport of Δ or an internal output commport of Δ.

property 6.12 *totcom is nonempty and prefix-closed*

For i/o-connectable components Γ and Δ and alphabet I,

(i) $\Gamma\ totcom_I\ \Delta$ is nonempty,

(ii) $\Gamma\ totcom_I\ \Delta$ is prefix-closed.

end of property

We present an alternative characterization of *totcom* in property 6.13.

property 6.13 *totcom*

Given are i/o-connectable components Γ and Δ and alphabet I. Let iobip F be such that $\mathbf{i}F = \mathbf{i}\Gamma \cap \mathbf{a}\Delta \cap I$ and $\mathbf{o}F = \mathbf{o}\Gamma \cap \mathbf{a}\Delta \cap I$,

$$\mathbf{t}(\Gamma\ totcom_I\ \Delta)$$
$$= \{s, t, u : (s \restriction \mathbf{a}\Gamma) \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge (u \restriction \mathbf{a}\Delta) \in \mathbf{t}(\mathbf{ptr}\,\Delta) \wedge s\,\mathbf{g}_F\,t \wedge t\,\mathbf{g}_F\,u : t\}$$

end of property

From definition 6.11, "***totcom***", definition 3.2, "i/o-connectable", and property 6.6 we infer the symmetry of ***totcom***, see property 6.14.

property 6.14 *totcom is symmetric*

For i/o-connectable components Γ and Δ and alphabet I,

$$\Gamma \, \textit{totcom}_I \, \Delta = \Delta \, \textit{totcom}_I \, \Gamma$$

end of property

We present some running examples to show applications of the composition method. Apart from this, the specific reason to include the running example that starts in example 6.15 is that the combination of trace structures $\mathbf{ptr}\,\Gamma_0$ and $\mathbf{ptr}\,\Delta_0$ reveals computation interference hazard at one of the external inputs.

example 6.15

We consider i/o-connectable components Γ_0 and Δ_0; in this example alphabet I equals $\{b\}$, see figure 6.7.

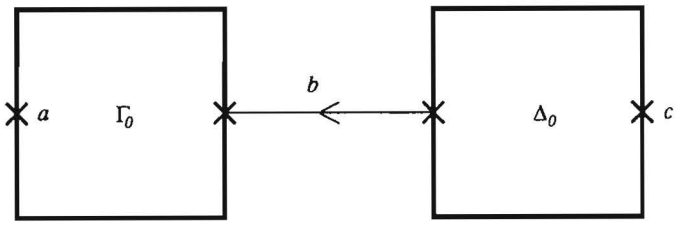

figure 6.7
Connection of components Γ_0 and Δ_0.

Components Γ_0 and Δ_0 are defined by:

$$\mathbf{o}\Gamma_0 \stackrel{\text{def}}{=} \{a\}, \quad \mathbf{i}\Gamma_0 \stackrel{\text{def}}{=} \{b\}, \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_0) \stackrel{\text{def}}{=} \mathbf{pref}\{ba\},$$
$$\mathbf{o}\Delta_0 \stackrel{\text{def}}{=} \{b\}, \quad \mathbf{i}\Delta_0 \stackrel{\text{def}}{=} \{c\}, \quad \mathbf{t}(\mathbf{ptr}\,\Delta_0) \stackrel{\text{def}}{=} \mathbf{pref}\{cbc\}.$$

The state graphs of components Γ_0 and Δ_0 are shown in figure 6.8.

figure 6.8a

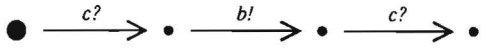

figure 6.8b

State graphs of components Γ_0 (figure 6.8a) and Δ_0 (figure 6.8b).

From definition 6.11, "***totcom***", we derive that

Γ_0 ***totcom***$_{\{b\}}\,\Delta_0$
$= \langle \{a, b, c\}, \{\varepsilon, c, cb, cc, cba, cbc, ccb, cbac, cbca, ccba\} \rangle$

Notice that symbol a in the traces in this trace structure occurs on account of definition 6.11(iii); symbols b and c in these traces occur on account of definition 6.11(iv). We present the state graph of Γ_0 ***totcom***$_{\{b\}}\,\Delta_0$ in figure 6.9.

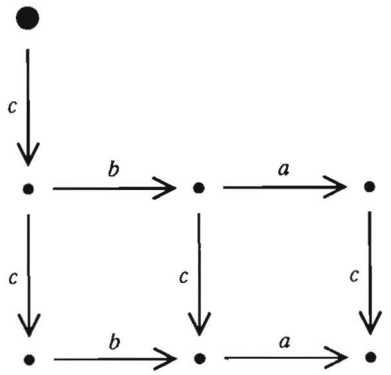

figure 6.9
State graph of trace structure Γ_0 ***totcom***$_{\{b\}}\,\Delta_0$.

We notice that $cc \in \mathbf{t}(\Gamma_0$ ***totcom***$_{\{b\}}\,\Delta_0)$ on account of definition 6.11(iv), since $c \in \mathbf{a}\Delta_0 \setminus \mathbf{o}\Gamma_0$, $c \in \mathbf{t}(\Gamma_0$ ***totcom***$_{\{b\}}\,\Delta_0)$, $cbc \in (\mathbf{a}\Gamma_0 \cup \mathbf{a}\Delta_0)^*$, $cbc \restriction \mathbf{a}\Delta_0 = cbc$, $cbc \in \mathbf{t}(\mathbf{ptr}\,\Delta_0)$, and $cc\,\mathbf{g}_{F_0}\,cbc$, for iobip F_0 such that $\mathbf{i}F_0 = \{b\}$ (i.e. $\mathbf{i}\Gamma_0 \cap \mathbf{a}\Delta_0 \cap \{b\}$) and $\mathbf{o}F_0 = \varnothing$ (i.e. $\mathbf{o}\Gamma_0 \cap \mathbf{a}\Delta_0 \cap \{b\}$).

end of example

The specific reason to include the running example that starts in example 6.16 is that the combination of trace structures $\mathbf{ptr}\,\Gamma_0$ and $\mathbf{ptr}\,\Delta_0$ reveals computation interference hazard at one of the internal inputs.

example 6.16

We consider i/o-connectable components Γ_I and Δ_I; in this example alphabet I equals $\{b\}$. Components Γ_I and Δ_I are defined by:

$$\mathbf{o}\Gamma_I \stackrel{\text{def}}{=} \{a\}, \quad \mathbf{i}\Gamma_I \stackrel{\text{def}}{=} \{b\}, \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_I) \stackrel{\text{def}}{=} \mathbf{pref}\{ab\},$$
$$\mathbf{o}\Delta_I \stackrel{\text{def}}{=} \{b\}, \quad \mathbf{i}\Delta_I \stackrel{\text{def}}{=} \{c\}, \quad \mathbf{t}(\mathbf{ptr}\,\Delta_I) \stackrel{\text{def}}{=} \mathbf{pref}\{cb\}.$$

The state graphs of components Γ_I and Δ_I are shown in figure 6.10.

figure 6.10a figure 6.10b

State graphs of components Γ_I (figure 6.10a) and Δ_I (figure 6.10b).

From definition 6.11, "***totcom***", we derive that

$$\Gamma_I\ \textit{totcom}_{\{b\}}\ \Delta_I = \langle \{a, b, c\}, \{\varepsilon, a, c, ac, ca, cb, acb, cab, cba\} \rangle$$

We present the state graph of $\Gamma_I\ \textit{totcom}_{\{b\}}\ \Delta_I$ in figure 6.11.

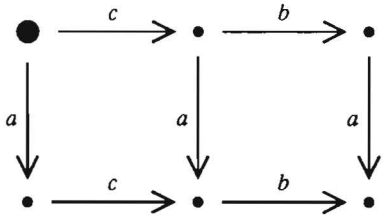

figure 6.11

State graph of trace structure $\Gamma_I\ \textit{totcom}_{\{b\}}\ \Delta_I$.

We notice that $cb \in \mathbf{t}(\Gamma_I\ \textit{totcom}_{\{b\}}\ \Delta_I)$ on account of definition 6.11(iv), since $b \in \mathbf{a}\Delta_I \setminus \mathbf{o}\Gamma_I$, $c \in \mathbf{t}(\Gamma_I\ \textit{totcom}_{\{b\}}\ \Delta_I)$, $cb \in (\mathbf{a}\Gamma_I \cup \mathbf{a}\Delta_I)^*$, $cb \upharpoonright \mathbf{a}\Delta_I = cb$, $cb \in \mathbf{t}(\mathbf{ptr}\,\Delta_I)$, and $cb\,\mathbf{g}_{F_I}\,cb$, for iobip F_I such that $\mathbf{i}F_I = \{b\}$ (i.e. $\mathbf{i}\Gamma_I \cap \mathbf{a}\Delta_I \cap \{b\}$) and $\mathbf{o}F_I = \varnothing$ (i.e. $\mathbf{o}\Gamma_I \cap \mathbf{a}\Delta_I \cap \{b\}$). On the other hand we notice that $ab \notin \mathbf{t}(\Gamma_I\ \textit{totcom}_{\{b\}}\ \Delta_I)$ on account of definition 6.11(iv), since $b \in \mathbf{o}\Delta_I$ and $\neg(ab\,\mathbf{g}_{F_I}\,u)$ for any trace u such that $u \upharpoonright (\mathbf{a}\Delta_I) \in \mathbf{t}(\mathbf{ptr}\,\Delta_I)$ because of $\#_c u = 0$.

end of example

In examples 6.15 and 6.16 we presented 'toy problems' to illustrate the calculation of the composite in this section. We present a more realistic case in the running example that starts in example 6.17: how to compose a Muller C-element out of a majority element and an asymmetric fork element.

example 6.17

We consider i/o-connectable components Γ_2 and Δ_2; in this example alphabet I equals the empty set $\varnothing$, see figure 6.12.

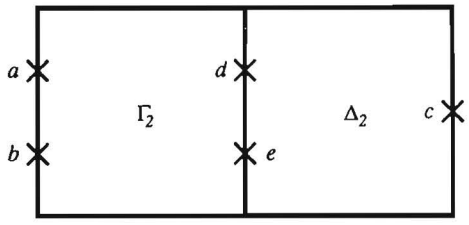

figure 6.12
Connection of components Γ_2 and Δ_2.

Component Γ_2 models a majority element, cf. example 2.52, where $\mathbf{i}\Gamma_2 = \{a, b, e\}$ and $\mathbf{o}\Gamma_2 = \{d\}$. Component Δ_2 models an asymmetric fork element, cf. example 2.49, where $\mathbf{i}\Delta_2 = \{d\}$, $\mathbf{o}\Delta_2 = \{e, c\}$, and c is associated with the commport that models the delayed output. From definition 6.11, "*totcom*", we derive Γ_2 *totcom*$_\varnothing$ Δ_2, see figure 6.13.

Notice that symbols a, b, and d in the traces in this trace structure occur on account of definition 6.11(iii); symbols c and e in these traces occur on account of definition 6.11(iv).

We notice that $abdeabd \in \mathbf{t}(\Gamma_2$ *totcom*$_\varnothing \Delta_2)$ on account of definition 6.11(iii), since $d \in \mathbf{a}\Gamma_2 \setminus \mathbf{o}\Delta_2$, $abdeab \in \mathbf{t}(\Gamma_2$ *totcom*$_\varnothing \Delta_2)$, $abdeabd \in (\mathbf{a}\Gamma_2 \cup \mathbf{a}\Delta_2)^*$, $abdeabd \restriction \mathbf{a}\Gamma_2 = abdeabd$, $abdeabd \in \mathbf{t}(\mathbf{ptr}\,\Gamma_2)$, and $abdeabd\, \mathbf{g}_{F_2}\, abdeabd$ for iobip F_2 such that $\mathbf{a}F_2 = \varnothing$.

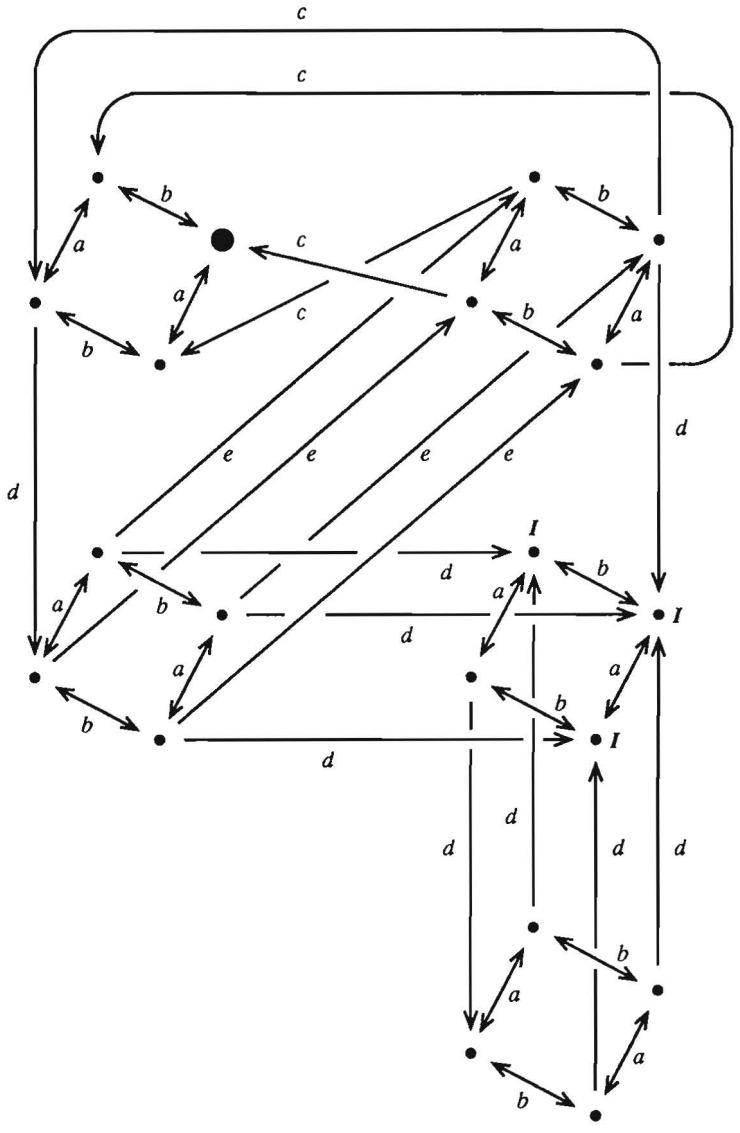

figure 6.13
State graph of trace structure Γ_2 *totcom*$_\varnothing$ Δ_2 .

Three states in the above state graph have been marked *I*; this marking will be explained later.

end of example

6.1.1 Absence of computation interference hazard

In this subsection we deal with the correctness concern absence of computation interference hazard. We do so in definition 6.24, "***totcomncih***", by reducing trace structure $\Gamma\, \mathit{totcom}_I \Delta$. In order to do this we introduce definition 6.18, "**cihi**". Trace set $\mathbf{cihi}_I(\Gamma, \Delta)$ consists of the traces of $\Gamma\, \mathit{totcom}_I \Delta$ that are associated with computation interference hazard at Δ.

definition 6.18 **cihi**

Given are i/o-connectable components Γ and Δ and alphabet I. Let iobip F be such that $\mathbf{i}F = \mathbf{i}\Gamma \cap \mathbf{a}\Delta \cap I$ and $\mathbf{o}F = \mathbf{o}\Gamma \cap \mathbf{a}\Delta \cap I$. We define trace set $\mathbf{cihi}_I(\Gamma, \Delta)$ by:

$$\begin{aligned}\mathbf{cihi}_I(\Gamma, \Delta) \stackrel{\text{def}}{=} \{ & a, t, u \\ & : a \in \mathbf{i}\Delta \wedge t \in \mathbf{t}(\Gamma\, \mathit{totcom}_I \Delta) \wedge u \in (\mathbf{a}\Gamma \cup \mathbf{a}\Delta)^* \\ & \quad \wedge (u \restriction \mathbf{a}\Delta) \in \mathbf{t}(\mathbf{ptr}\,\Delta) \wedge (ua \restriction \mathbf{a}\Delta) \notin \mathbf{t}(\mathbf{ptr}\,\Delta) \wedge t\, \mathbf{g}_F\, ua \\ & : t \\ & \}\end{aligned}$$

end of definition

We continue by calculating the **cihi** for our running examples. In example 6.19 we present an example in which there is computation interference hazard at an external input.

example 6.19

We consider components Γ_0 and Δ_0, see example 6.15. From definition 6.18, "**cihi**", we derive that

$\mathbf{cihi}_{\{b\}}(\Gamma_0, \Delta_0) = \{cc\}$ and

$\mathbf{cihi}_{\{b\}}(\Delta_0, \Gamma_0) = \varnothing$.

Trace cc is an element of $\mathbf{cihi}_{\{b\}}(\Gamma_0, \Delta_0)$, since $c \in \mathbf{i}\Delta_0$, $cc \in \mathbf{t}(\Gamma_0\, \mathit{totcom}_{\{b\}} \Delta_0)$, $c \in (\mathbf{a}\Gamma_0 \cup \mathbf{a}\Delta_0)^*$, $c \restriction \mathbf{a}\Delta_0 = c$, $c \in \mathbf{t}(\mathbf{ptr}\,\Delta_0)$, $cc \restriction \mathbf{a}\Delta_0 = cc$, $cc \notin \mathbf{t}(\mathbf{ptr}\,\Delta_0)$, and $cc\, \mathbf{g}_{F_0}\, cc$.

end of example

Example 6.20 exhibits computation interference hazard at an internal input.

example 6.20

We consider components Γ_1 and Δ_1, see example 6.16. From definition 6.18, "**cihi**", we derive that

$$\mathbf{cihi}_{\{b\}}(\Gamma_1, \Delta_1) = \varnothing$$

$$\mathbf{cihi}_{\{b\}}(\Delta_1, \Gamma_1) = \{cb\}$$

Let iobip F_7 be such that $F_7 = \overline{F_1}$. Trace cb is an element of $\mathbf{cihi}_{\{b\}}(\Delta_1, \Gamma_1)$, since $b \in \mathbf{i}\Gamma_1$, $cb \in \mathbf{t}(\Gamma_1\ totcom_{\{b\}}\ \Delta_1)$, $c \in (\mathbf{a}\Gamma_1 \cup \mathbf{a}\Delta_1)^*$, $c \restriction \mathbf{a}\Gamma_1 = \varepsilon$, $\varepsilon \in \mathbf{t}(\mathbf{ptr}\,\Gamma_1)$, $cb \restriction \mathbf{a}\Gamma_1 = b$, $b \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_1)$, and $cb\,\mathbf{g}_{F_7}\,cb$.

end of example

Like example 6.20, example 6.21 exhibits computation interference hazard at an internal input.

example 6.21

We consider components Γ_2 and Δ_2 and iobip F_2, see example 6.17. The majority element accepts all commsigs that it receives, see example 2.52; we derive from definition 6.18, "**cihi**", that $\mathbf{cihi}_{\varnothing}(\Delta_2, \Gamma_2) = \varnothing$.

We notice that $(\mathbf{A}\, t, u : tdud \in \mathbf{t}(\mathbf{ptr}\,\Delta_2) \wedge (\#_d u = 0) : u = ec)$; as a consequence, $ded \notin \mathbf{t}(\mathbf{ptr}\,\Delta_2)$. Since $abdeabd \in \mathbf{t}(\Gamma_2\ totcom_{\varnothing}\ \Delta_2)$ and $ded = abdeabd \restriction \mathbf{a}\Delta_2$, there is computation interference hazard at Δ_2. As a consequence, trace set $\mathbf{cihi}_{\varnothing}(\Gamma_2, \Delta_2)$ is nonempty; it consists of all traces that lead from the initial state via states that have not been marked to a state marked ***I*** in the diagram of the state graph of $\Gamma_2\ totcom_{\varnothing}\ \Delta_2$ shown in figure 6.13, see example 6.17. From definition 6.18, "**cihi**", we infer that $abdeabd \in \mathbf{cihi}_{\varnothing}(\Gamma_2, \Delta_2)$, since $d \in \mathbf{i}\Delta_2$, $abdeabd \in \mathbf{t}(\Gamma_2\ totcom_{\varnothing}\ \Delta_2)$, $abdeab \in (\mathbf{a}\Gamma_2 \cup \mathbf{a}\Delta_2)^*$, $abdeab \restriction \mathbf{a}\Delta_2 = de$, $de \in \mathbf{t}(\mathbf{ptr}\,\Delta_2)$, $abdeabd \restriction \mathbf{a}\Delta_2 = ded$, $ded \notin \mathbf{t}(\mathbf{ptr}\,\Delta_2)$, and $abdeabd\,\mathbf{g}_{F_2}\,abdeabd$.

end of example

In definition 6.24, "***totcomncih***", we use the technique "transformation into computation interference hazard", see subsection 3.3.0; here we transform "computation interference hazard at the boundary of Γ or Δ" into "computation interference hazard at the external boundary of the composite of Γ and Δ". We have argued in subsection 3.3.0 that there may be initial problems when modeling correctness concerns in our Communication Model. We do not want to end up with a component that has an empty trace set, when reducing trace structure $\Gamma\, totcom_I\, \Delta$ to trace structure $\Gamma\, totcomncih_I\, \Delta$. For this reason we define predicate $\Gamma\, NICIH_I\, \Delta$, see definition 6.22.

definition 6.22 ***NICIH***

For i/o-connectable components Γ and Δ and alphabet I, we define predicate $\Gamma\, NICIH_I\, \Delta$ by:

$$\Gamma\, NICIH_I\, \Delta \stackrel{\text{def}}{=} (\mathbf{A}\, s : s \in (\mathbf{cihi}_I(\Gamma, \Delta) \cup \mathbf{cihi}_I(\Delta, \Gamma)) : I(s \restriction \mathbf{extinp}(\Gamma, \Delta)) > 0)$$

end of definition

Notice that if $\neg(\Gamma\, NICIH_I\, \Delta)$, then the composite of Γ and Δ, where I is associated with the indirect connection, has computation interference hazard: computation interference can occur before any external input has occurred. I/o-connectable components Γ and Δ can be connected under alphabet I with *no initial computation interference hazard* if and only if $\Gamma\, NICIH_I\, \Delta$.

From the symmetry of **extinp** we infer the symmetry of $NICIH_I$.

property 6.23 ***NICIH is symmetric***

For i/o-connectable components Γ and Δ and alphabet I,

$$\Gamma\, NICIH_I\, \Delta = \Delta\, NICIH_I\, \Gamma$$

end of property

Now we are ready to reduce $\Gamma\, totcom_I\, \Delta$ to $\Gamma\, totcomncih_I\, \Delta$.

definition 6.24 ***totcomncih***

For i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\, NICIH_I\, \Delta$, trace structure $\Gamma\, totcomncih_I\, \Delta$ is defined by:

$$\Gamma\, totcomncih_I\, \Delta \stackrel{\text{def}}{=} \mathbf{redts}(\Gamma\, totcom_I\, \Delta, \mathbf{extinp}(\Gamma, \Delta), \mathbf{cihi}_I(\Gamma, \Delta) \cup \mathbf{cihi}_I(\Delta, \Gamma))$$

end of definition

The prefix-closedness of $\Gamma\,\textit{totcomncih}_I\,\Delta$ follows from property 6.12(ii) and property 1.39. The nonemptiness of $\Gamma\,\textit{totcomncih}_I\,\Delta$ follows from property 1.40, using $\Gamma\,\textit{NICIH}_I\,\Delta$ and property 6.12(i).

From property 6.23, "***NICIH*** is symmetric", and property 6.14, "***totcom*** is symmetric", we infer the symmetry of ***totcomncih***.

property 6.25 ***totcomncih is symmetric***

For i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\,\textit{NICIH}_I\,\Delta$,

$$\Gamma\,\textit{totcomncih}_I\,\Delta = \Delta\,\textit{totcomncih}_I\,\Gamma$$

end of property

We now calculate ***totcomncih*** for our running examples. We show that by calculating ***totcomncih*** we have dealt with computation interference hazard.

example 6.26

We consider components Γ_0 and Δ_0, see example 6.15 and example 6.19. From definition 6.24, "***totcomncih***", we derive that

$$\Gamma_0\,\textit{totcomncih}_{\{b\}}\,\Delta_0 = \langle\{a, b, c\}, \{\varepsilon, c, cb, cba, cbc, cbac, cbca\}\rangle$$

see figure 6.14.

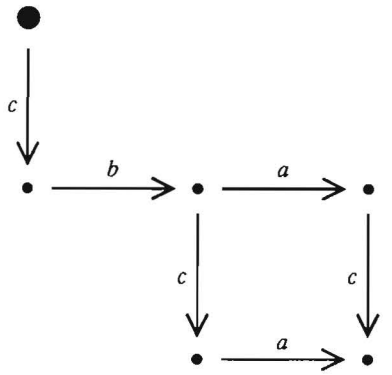

figure 6.14
State graph of trace structure $\Gamma_0\,\textit{totcomncih}_{\{b\}}\,\Delta_0$.

Since $cc \in \mathbf{cihi}_{\{b\}}(\Gamma_0, \Delta_0)$, we infer that $cc \notin \mathbf{t}(\Gamma_0\,\textit{totcomncih}_{\{b\}}\,\Delta_0)$. In subsection 6.1.2 we will deal with hiding the internal communication, while maintaining absence of computation interference hazard.

end of example

example 6.27

We consider components Γ_l and Δ_l, see example 6.16 and example 6.20. From definition 6.24, "***totcomncih***", we derive that

$$\Gamma_l \; \textbf{\textit{totcomncih}}_{\{b\}} \Delta_l = < \{a, b, c\}, \{\varepsilon, a, ac, acb\} >$$

see figure 6.15.

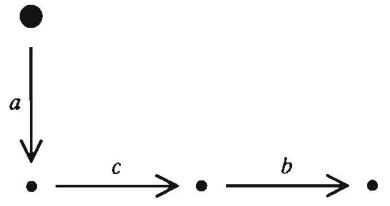

figure 6.15
State graph of trace structure Γ_l ***totcomncih***$_{\{b\}} \Delta_l$.

Since $cb \in \textbf{cihi}_{\{b\}}(\Delta_l, \Gamma_l)$, we infer that $cb \notin \textbf{t}(\Gamma_l \; \textbf{\textit{totcomncih}}_{\{b\}} \Delta_l)$. Since $b \notin \textbf{extinp}(\Gamma_l, \Delta_l)$ and $c \in \textbf{extinp}(\Gamma_l, \Delta_l)$, we see that not only trace cb has been removed while calculating Γ_l ***totcomncih***$_{\{b\}} \Delta_l$, but also trace c.

end of example

example 6.28

We consider components Γ_2 and Δ_2, see example 6.17 and example 6.21. From definition 6.24, "***totcomncih***", we derive trace structure Γ_2 ***totcomncih***$_{\varnothing}$ Δ_2, see figure 6.16.

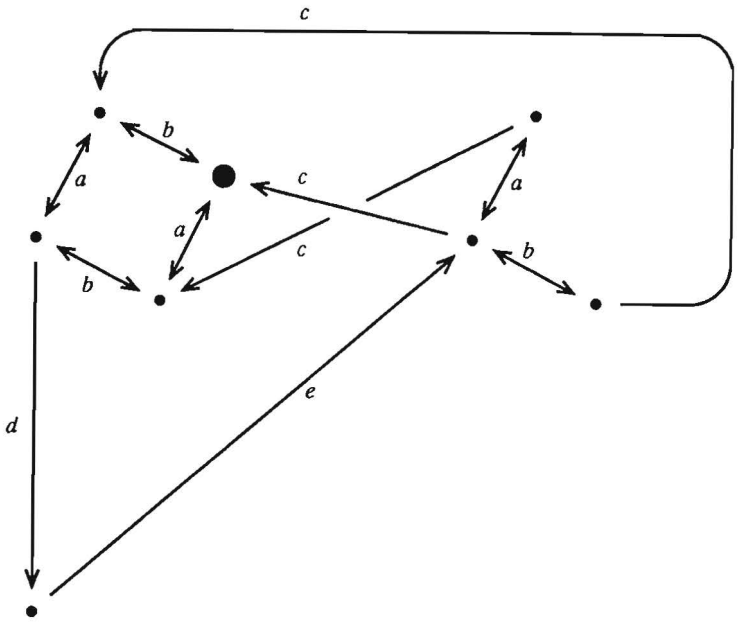

figure 6.16
State graph of trace structure Γ_2 ***totcomncih***$_{\varnothing}$ Δ_2.

In the state graph shown in figure 6.16 all states have been located at relative positions that are equal to those in figure 6.13. The state graph in figure 6.16 has been redrawn in figure 6.17.

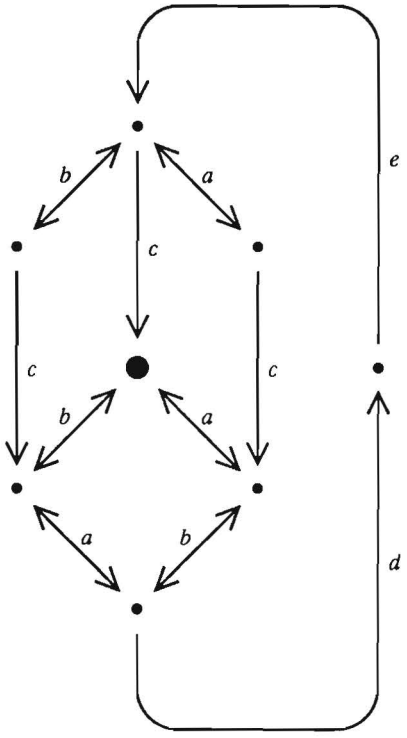

figure 6.17
State graph of trace structure Γ_2 ***totcomncih***$_{\varnothing}\Delta_2$.

Since $abdeabd \in \mathbf{cihi}_{\varnothing}(\Delta_2, \Gamma_2)$, we infer that $abdeabd \notin \mathbf{t}(\Gamma_2$ ***totcomncih***$_{\varnothing}\Delta_2)$. Since $d \notin \mathbf{extinp}(\Gamma_2, \Delta_2)$ and $b \in \mathbf{extinp}(\Gamma_2, \Delta_2)$, we see that not only trace *abdeabd* has been removed while calculating Γ_2 ***totcomncih***$_{\varnothing}\Delta_2$, but also trace *abdeab*.

end of example

6.1.2 Hiding the internal communication

The last step in the construction of the composite is "hiding the internal communication". In the beginning of section 6.1 we have argued that the composite has to be maximal. Since absence of computation interference hazard is our correctness concern, projecting onto the external alphabet is sufficient as far as external outputs are concerned, see definition 6.29(iii); however, an additional restriction with respect to external inputs is needed, see definition 6.29(iv).

definition 6.29 ***extcomncih***

For i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\ \boldsymbol{NICIH}_I\ \Delta$, trace structure $\Gamma\ \boldsymbol{extcomncih}_I\ \Delta$ is defined by:

(i) $\mathbf{a}(\Gamma\ \boldsymbol{extcomncih}_I\ \Delta) \stackrel{\text{def}}{=} \mathbf{a}\Gamma \div \mathbf{a}\Delta$

(ii) $\varepsilon \in \mathbf{t}(\Gamma\ \boldsymbol{extcomncih}_I\ \Delta)$

(iii) for trace u and symbol f such that $u \in \mathbf{t}(\Gamma\ \boldsymbol{extcomncih}_I\ \Delta)$, $f \in \mathbf{extoutp}(\Gamma, \Delta)$, and $(\mathbf{E}\, t : t \in \mathbf{t}(\Gamma\ \boldsymbol{totcomncih}_I\ \Delta) : t \restriction (\mathbf{a}\Gamma \div \mathbf{a}\Delta) = uf)$,

$$uf \in \mathbf{t}(\Gamma\ \boldsymbol{extcomncih}_I\ \Delta)$$

(iv) for trace u and symbol g such that $u \in \mathbf{t}(\Gamma\ \boldsymbol{extcomncih}_I\ \Delta)$, $g \in \mathbf{extinp}(\Gamma, \Delta)$, and $(\mathbf{A}\, t : t \in \mathbf{t}(\Gamma\ \boldsymbol{totcomncih}_I\ \Delta) \wedge (t \restriction (\mathbf{a}\Gamma \div \mathbf{a}\Delta) = u) : tg \in \mathbf{t}(\Gamma\ \boldsymbol{totcomncih}_I\ \Delta))$,

$$ug \in \mathbf{t}(\Gamma\ \boldsymbol{extcomncih}_I\ \Delta)$$

(v) completeness axiom: $\mathbf{t}(\Gamma\ \boldsymbol{extcomncih}_I\ \Delta)$ contains no traces that are not required by (ii), (iii), or (iv).

end of definition

By $\Gamma\ \boldsymbol{extcomncih}_I\ \Delta$ we denote the trace structure that is associated with the *external communication of the composite of* Γ *and* Δ *under* I *without computation interference hazard*. Every trace in the trace structure of this composite must belong to the projection of trace set $\mathbf{t}(\Gamma\ \boldsymbol{totcomncih}_I\ \Delta)$ onto the external alphabet (i.e. $\mathbf{a}\Gamma \div \mathbf{a}\Delta$). This gives un upper limit for trace set $\mathbf{t}(\Gamma\ \boldsymbol{extcomncih}_I\ \Delta)$. In order to maintain absence of computation interference hazard, there is an additional restriction needed for external inputs; this is why restriction $(\mathbf{A}\, t : t \in \mathbf{t}(\Gamma\ \boldsymbol{totcomncih}_I\ \Delta) \wedge (t \restriction (\mathbf{a}\Gamma \div \mathbf{a}\Delta) = u) : tg \in \mathbf{t}(\Gamma\ \boldsymbol{totcomncih}_I\ \Delta))$, occurs in definition 6.29(iv). There is a universal quantification in this restriction, since we deal not only with absence of some instance of computation interference but with absence of computation interference hazard: we have to guarantee that computation interference does not occur.

In definition 6.29, "***extcomncih***", we construct $extcomncih_I$ from $totcomncih_I$. This construction can be interpreted as 'projection onto the external alphabet of the composite under invariance of absence of computation interference hazard': we may interpret $\Gamma\, totcomncih_I\, \Delta$ as the communication behavior of the composite of Γ and Δ under I at the curly boundary in figure 6.18a.

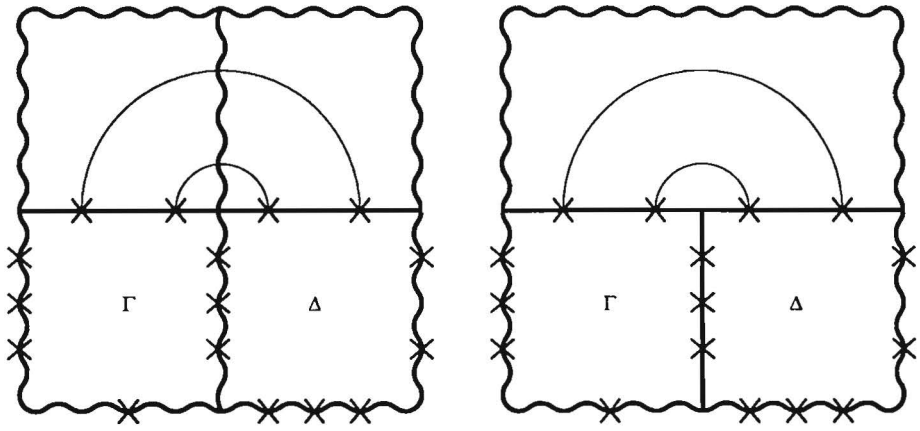

figure 6.18a figure 6.18b

Interpretation of trace structures $\Gamma\, totcomncih_I\, \Delta$ (6.18a) and $\Gamma\, extcomncih_I\, \Delta$ (6.18b).

We interpret $\Gamma\, extcomncih_I\, \Delta$ as the communication behavior of the composite of Γ and Δ under I at the curly boundary in figure 6.18b.

From property 6.23, "***NICIH*** is symmetric", property 6.25, "***totcomncih*** is symmetric", and definition 6.29, "***extcomncih***", we infer the symmetry of ***extcomncih***.

property 6.30 ***extcomncih*** *is symmetric*

For i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\, NICIH_I\, \Delta$,

$$\Gamma\, extcomncih_I\, \Delta = \Delta\, extcomncih_I\, \Gamma$$

end of property

remark 6.31

Notice that the technique that we use in definition 6.29, "***extcomncih***", is generally applicable when hiding internal communication. Since we only use it once in this monograph we have not chosen to present it as a general technique; we only present the one instantiation of it.

end of remark

We now hide the internal communication in our running examples by calculating ***extcomncih*** for them.

example 6.32

We consider components Γ_0 and Δ_0, see example 6.15 and example 6.26. From definition 6.29, "***extcomncih***", we derive that

$$\Gamma_0 \, \textit{extcomncih}_{\{b\}} \Delta_0 = < \{a, c\}, \{\varepsilon, c, ca, cac\} >$$

Notice that by calculating $\Gamma_0 \, \textit{extcomncih}_{\{b\}} \Delta_0$ we have 'lost' trace cc $(= cbc \restriction (\mathbf{a}\Gamma_0 \div \mathbf{a}\Delta_0))$ on account of definition 6.29(iv), since $c \in \mathbf{t}(\Gamma_0 \, \textit{totcomncih}_{\{b\}} \Delta_0)$ and $cc \notin \mathbf{t}(\Gamma_0 \, \textit{totcomncih}_{\{b\}} \Delta_0)$.

In subsection 6.1.3 we will interpret trace structure $\Gamma_0 \, \textit{extcomncih}_{\{b\}} \Delta_0$ in the Communication Model: we will define a component that has this trace structure.

end of example

example 6.33

We consider components Γ_1 and Δ_1, see example 6.16 and example 6.27. From definition 6.29, "***extcomncih***", we derive that

$$\Gamma_1 \, \textit{extcomncih}_{\{b\}} \Delta_1 = < \{a, c\}, \{\varepsilon, a, ac\} >$$

end of example

example 6.34

We consider components Γ_2 and Δ_2, see example 6.17 and example 6.28. From definition 6.29, "***extcomncih***", we derive trace structure $\Gamma_2 \, \textit{extcomncih}_{\varnothing} \Delta_2$, see figure 6.19.

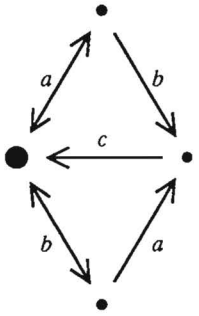

figure 6.19
State graph of trace structure $\Gamma_2 \, \textit{extcomncih}_{\varnothing} \Delta_2$.

Notice that by calculating $\Gamma_2 \, \textit{extcomncih}_{\varnothing} \Delta_2$ we have 'lost' –among others– trace abb on account of definition 6.29(iv), since $abd \in \mathbf{t}(\Gamma_2 \, \textit{totcomncih}_{\varnothing} \Delta_2)$, $abdb \notin \mathbf{t}(\Gamma_2 \, \textit{totcomncih}_{\varnothing} \Delta_2)$, and $abb = abdb \restriction (\mathbf{a}\Gamma_2 \div \mathbf{a}\Delta_2)$.

end of example

6.1.3 Composite of two components

The component that is *the composite of* Γ *and* Δ *under* I *without computation interference hazard* is denoted by $\Gamma\,\boldsymbol{COMPNCIH}_I\Delta$, see definition 6.35. At this point "composition" becomes defined in our Communication Model. In our Communication Model we are only interested in composites without computation interference hazard.

definition 6.35 ***COMPNCIH***

For i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\,\boldsymbol{NICIH}_I\Delta$, component $\Gamma\,\boldsymbol{COMPNCIH}_I\Delta$ is defined by:

$$\mathbf{i}(\Gamma\,\boldsymbol{COMPNCIH}_I\Delta) \stackrel{\text{def}}{=} \mathbf{extinp}(\Gamma, \Delta)$$

$$\mathbf{o}(\Gamma\,\boldsymbol{COMPNCIH}_I\Delta) \stackrel{\text{def}}{=} \mathbf{extoutp}(\Gamma, \Delta)$$

$$\mathbf{ptr}(\Gamma\,\boldsymbol{COMPNCIH}_I\Delta) \stackrel{\text{def}}{=} \Gamma\,\boldsymbol{extcomncih}_I\Delta$$

end of definition

From property 6.23, "***NICIH*** is symmetric", property 6.30, "***extcomncih*** is symmetric", and the symmetry of i/o-connectability, **extinp**, and **extoutp**, we infer the symmetry of ***COMPNCIH***.

property 6.36 ***COMPNCIH*** *is symmetric*

For i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\,\boldsymbol{NICIH}_I\Delta$,

$$\Gamma\,\boldsymbol{COMPNCIH}_I\Delta = \Delta\,\boldsymbol{COMPNCIH}_I\Gamma$$

end of property

Although the proof of the associativity of ***COMPNCIH*** is very long and awkward, there is nothing to be learned from it; for this reason we present property 6.37 without a proof.

property 6.37 ***COMPNCIH*** *is associative*

For alphabet I and components Γ, Δ, and Θ such that each pair of them is i/o-connectable,

(i) $(\Gamma\,\boldsymbol{COMPNCIH}_I\Delta)\,\boldsymbol{NICIH}_I\Theta = \Gamma\,\boldsymbol{NICIH}_I(\Delta\,\boldsymbol{COMPNCIH}_I\Theta)$

(ii) $(\Gamma\,\boldsymbol{COMPNCIH}_I\Delta)\,\boldsymbol{COMPNCIH}_I\Theta = \Gamma\,\boldsymbol{COMPNCIH}_I(\Delta\,\boldsymbol{COMPNCIH}_I\Theta)$

end of property

In property 6.37 (i) either the left and right hand side both hold or each of them "either is not defined or does not hold". In property 6.37 (ii) either the left and right hand side both are defined or neither is defined; if both are defined then they are equal.

We now are able to interpret the composites that have been constructed in our running examples.

example 6.38

We consider components Γ_0 and Δ_0, see example 6.15 and example 6.32. From definition 6.35, "***COMPNCIH***", we derive component Γ_0 ***COMPNCIH***$_{\{b\}}\Delta_0$, see figure 6.20.

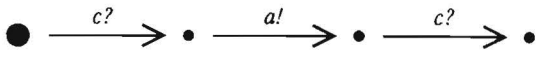

figure 6.20
State graph of component Γ_0 ***COMPNCIH***$_{\{b\}}\Delta_0$.

In example 6.15 we have seen that $cc \in \mathbf{t}(\Gamma_0$ ***totcom***$_{\{b\}}\Delta_0)$ and $cbc \in \mathbf{t}(\Gamma_0$ ***totcom***$_{\{b\}}\Delta_0)$, whereas $cc \notin \mathbf{t}(\mathbf{ptr}\,\Delta_0)$, but $cbc \in \mathbf{t}(\mathbf{ptr}\,\Delta_0)$. Since $cc = cbc \restriction \mathbf{a}\Delta_0$, we conclude that absence of computation interference hazard was not dealt with by calculating Γ_0 ***totcom***$_{\{b\}}\Delta_0$. After component Γ_0 ***COMPNCIH***$_{\{b\}}\Delta_0$ has accepted the commsig to which the first c is associated, it may or may not accept the commsig to which the second c is associated. For this reason, the environment of this composite has to postpone the sending of the latter commsig until it has received the commsig to which a is associated. After component Γ_0 ***COMPNCIH***$_{\{b\}}\Delta_0$ has sent the commsig to which a is associated, it accepts the commsig to which the second c is associated.

end of example

example 6.39

We consider components Γ_l and Δ_l, see example 6.16 and example 6.33. From definition 6.35, "***COMPNCIH***", we derive component Γ_l ***COMPNCIH***$_{\{b\}}$ Δ_l, see figure 6.21.

figure 6.21
State graph of component Γ_l ***COMPNCIH***$_{\{b\}}$ Δ_l.

In example 6.16 we have seen that $cb \in \mathbf{t}(\Gamma_l$ ***totcom***$_{\{b\}}$ $\Delta_l)$, whereas $b \notin \mathbf{t}(\mathbf{ptr}\,\Gamma_l)$. Since $b = cb\restriction \mathbf{a}\Gamma_l$, we conclude that absence of computation interference hazard was not dealt with by calculating Γ_l ***totcom***$_{\{b\}}$ Δ_l. We conclude that the environment of the composite Γ_l ***COMPNCIH***$_{\{b\}}$ Δ_l should not send a commsig to which c is associated until it has received a commsig to which a is associated, since after the sending of the latter commsig it is guaranteed that internally (between Γ_l and Δ_l) no computation interference occurs.

In this example hiding the internal communication (alphabet $\{b\}$) has been no problem: a simple projection onto the external alphabet has been sufficient.

end of example

example 6.40

We consider components Γ_2 and Δ_2, see example 6.17 and example 6.34. From definition 6.35, "***COMPNCIH***", we derive component Γ_2 ***COMPNCIH***$_\varnothing$ Δ_2, see figure 6.22.

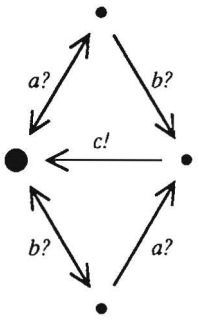

figure 6.22
State graph of component Γ_2 ***COMPNCIH***$_\varnothing$ Δ_2.

Component Γ_2 ***COMPNCIH***$_\varnothing$ Δ_2 is equal to Ebergen's Γ_{ncel}, see [Ebergen 87]. From example 5.38, we infer that **DIE**(Γ_2 ***COMPNCIH***$_\varnothing$ Δ_2) = Γ_c, cf. example 2.48. From example 5.51, we infer that **CBDI**(Γ_2 ***COMPNCIH***$_\varnothing$ Δ_2) = Γ_c.

end of example

In property 6.41 we present the unity element of the composition operator ***COMPNCIH***.

property 6.41 *unity element of* ***COMPNCIH***$_I$

Given is component Γ. Let component Δ be such that $\mathbf{a}\Delta = \varnothing$ and $\mathbf{t}(\mathbf{ptr}\,\Delta) = \{\varepsilon\}$.

(i) Γ and Δ are i/o-connectable

(ii) for alphabet I, Γ ***NICIH***$_I$ Δ

(iii) for alphabet I, Γ ***COMPNCIH***$_I$ $\Delta = \Gamma$

and Δ ***COMPNCIH***$_I$ $\Gamma = \Gamma$

end of property

We notice that the unity element of ***COMPNCIH***$_I$ does not depend on alphabet I.

In property 6.41 we give conditions under which the trace structure of the composite of two components is equal to the blend, see definition 1.29, of the trace structures of the two components.

property 6.42

For i/o-connectable components Γ and Δ such that $\mathbf{cihi}_{\varnothing}(\Gamma,\Delta)=\varnothing$ and $\mathbf{cihi}_{\varnothing}(\Delta,\Gamma)=\varnothing$,

$$\Gamma \textit{extcomncih}_{\varnothing}\Delta = (\mathbf{ptr}\,\Gamma)\,\mathbf{b}(\mathbf{ptr}\,\Delta)$$

end of property

We investigate the distribution of **DSE** over ***COMPNCIH***. We first look at example 6.43.

example 6.43

We consider components Γ_3 and Δ_3; they are defined by:

$$\mathbf{o}\Gamma_3 \stackrel{\text{def}}{=} \{a,b\}, \quad \mathbf{i}\Gamma_3 \stackrel{\text{def}}{=} \{c\}, \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_3) \stackrel{\text{def}}{=} \mathbf{pref}\{cab\},$$
$$\mathbf{o}\Delta_3 \stackrel{\text{def}}{=} \varnothing, \quad \mathbf{i}\Delta_3 \stackrel{\text{def}}{=} \{a,b\}, \quad \mathbf{t}(\mathbf{ptr}\,\Delta_3) \stackrel{\text{def}}{=} \mathbf{pref}\{ab\}.$$

From definition 4.36, "**dse**", we derive that $\mathbf{t}(\mathbf{dse}\,\Gamma_3) = \{\varepsilon, c, ca, cb, cab, cba\}$ and $\mathbf{t}(\mathbf{dse}\,\Delta_3) = \{\varepsilon, a\}$. From definition 6.22, "***NICIH***", we derive that $\Gamma_3\, \textit{NICIH}_{\varnothing}\Delta_3$ and $(\mathbf{DSE}\,\Gamma_3)\textit{NICIH}_{\varnothing}(\mathbf{DSE}\,\Delta_3)$. Using theorem 4.45, "delay-safe enclosure", we infer from definition 6.35, "***COMPNCIH***", that $\mathbf{ptr}((\mathbf{DSE}\,\Gamma_3)\textit{COMPNCIH}_{\varnothing}(\mathbf{DSE}\,\Delta_3)) = \langle\{c\},\{\varepsilon\}\rangle$ and $\mathbf{ptr}(\Gamma_3\, \textit{COMPNCIH}_{\varnothing}\Delta_3) = \langle\{c\},\{\varepsilon, c\}\rangle$. From definition 4.36, "**dse**", we derive that $\mathbf{dse}(\Gamma_3\, \textit{COMPNCIH}_{\varnothing}\Delta_3) = \langle\{c\},\{\varepsilon, c\}\rangle$.

end of example

remark 6.44

From example 6.43 we see that, in general, for i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\, \textit{NICIH}_I\Delta$,

$$\mathbf{DSE}(\Gamma\, \textit{COMPNCIH}_I\Delta) \neq (\mathbf{DSE}\,\Gamma)\textit{COMPNCIH}_I(\mathbf{DSE}\,\Delta)$$

We conclude that, in general, **DSE** does not distribute over $\textit{COMPNCIH}_I$. A sufficient condition for the distribution of **DSE** over ***COMPNCIH*** is given in the following property.

end of remark

property 6.45

For i/o-connectable components Γ and Δ and alphabet I such that $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \subseteq I$ and $\Gamma\, \textit{NICIH}_I\Delta$,

$$\mathbf{DSE}(\Gamma\, \textit{COMPNCIH}_I\Delta) = (\mathbf{DSE}\,\Gamma)\textit{COMPNCIH}_I(\mathbf{DSE}\,\Delta)$$

end of property

From trace theory, see [Kaldewaij86], we know that $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \subseteq I$ is the condition that is needed for the distribution of projection over weaving of trace structures, i.e. $((\mathbf{ptr}\,\Gamma)\,\mathbf{w}\,(\mathbf{ptr}\,\Delta))\restriction I = (\mathbf{ptr}\,\Gamma\restriction I)\,\mathbf{w}\,(\mathbf{ptr}\,\Delta\restriction I)$. Analogously to property 6.45 we find property 6.46.

property 6.46

For i/o-connectable components Γ and Δ and alphabet I such that $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \subseteq I$ and $\Gamma\ \mathit{NICIH}_I\ \Delta$,

$$\mathbf{CBDS}(\Gamma\ \mathit{COMPNCIH}_I\ \Delta) = (\mathbf{CBDS}\,\Gamma)\,\mathit{COMPNCIH}_I\,(\mathbf{CBDS}\,\Delta)$$

end of property

6.1.4 Examples

In example 6.47 we show that computation interference hazard may be absent in the composite of components that have a mixed connection, whereas it is present in the composite of these components when they have an indirect connection.

example 6.47

We consider components Γ_4 and Δ_4; they are defined by:

$$\mathbf{o}\Gamma_4 \stackrel{\text{def}}{=} \{a,b\}, \quad \mathbf{i}\Gamma_4 \stackrel{\text{def}}{=} \{c,d\}, \quad \mathbf{t}(\mathbf{ptr}\,\Gamma_4) \stackrel{\text{def}}{=} \mathbf{pref}\{bad\},$$
$$\mathbf{o}\Delta_4 \stackrel{\text{def}}{=} \{c,d\}, \quad \mathbf{i}\Delta_4 \stackrel{\text{def}}{=} \{a,b\}, \quad \mathbf{t}(\mathbf{ptr}\,\Delta_4) \stackrel{\text{def}}{=} \mathbf{pref}\{abc,bad\}.$$

We consider the composition of Γ_4 and Δ_4 when they have an indirect connection, see figure 6.23.

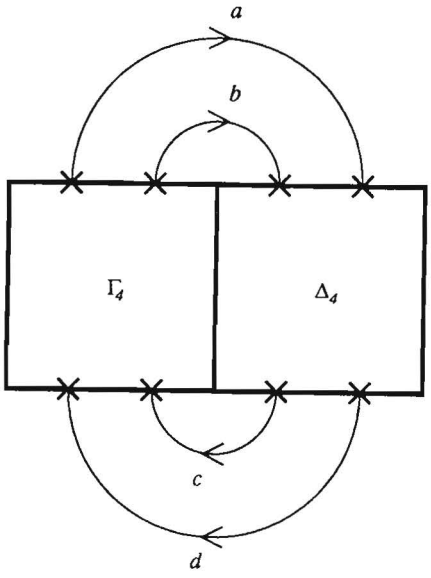

figure 6.23
Indirect connection of components Γ_4 and Δ_4.

We infer that $\Gamma_4\ \textit{totcom}_{\{a,b,c,d\}}\Delta_4 = \mathbf{pref}\{abc, abd, bac, bad\}$ from definition 6.11, "*totcom*". Using definition 6.18, "**cihi**", we find that $\mathbf{cihi}_{\{a,b,c,d\}}(\Gamma_4,\Delta_4)=\varnothing$ and $\mathbf{cihi}_{\{a,b,c,d\}}(\Delta_4,\Gamma_4)=\{abc,bac\}$. From definition 6.22, "*NICIH*", we derive that $\neg(\Gamma_4\ \textit{NICIH}_{\{a,b,c,d\}}\Delta_4)$.

We now consider the composition of Γ_4 and Δ_4 when they have a mixed connection, see figure 6.24.

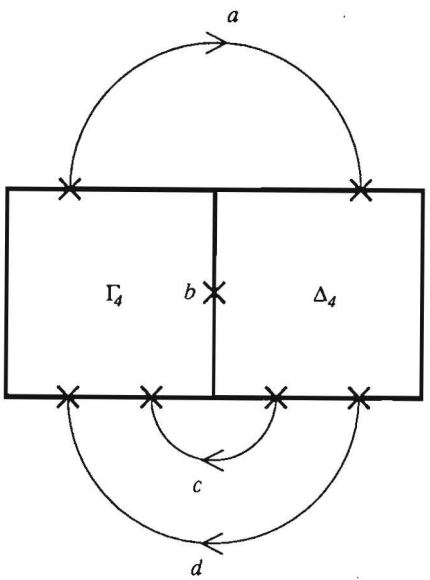

figure 6.24
Mixed connection of components Γ_4 and Δ_4.

Now, we infer that Γ_4 ***totcom***$_{\{a,c,d\}}\Delta_4 =$ **pref** $\{bad\}$. Furthermore, we find that **cihi**$_{\{a,c,d\}}(\Gamma_4, \Delta_4)=\varnothing$ and also **cihi**$_{\{a,c,d\}}(\Delta_4, \Gamma_4)=\varnothing$. As a consequence, we conclude that Γ_4 ***NICIH***$_{\{a,c,d\}}\Delta_4$. From definition 6.22, "***NICIH***", using property 1.42 we derive that $\mathbf{t}(\Gamma_4$ ***totcomncih***$_{\{a,c,d\}}\Delta_4)=$ **pref** $\{bad\}$.

We conclude that the problem with computation interference hazard in the composition with the indirect connection is not present in the composition with the mixed connection. Furthermore, we notice that it hasn't been necessary to confine the connection of Γ_4 and Δ_4 to be direct. Only the commports to which b is associated are directly connected, all other commports are indirectly connected.

end of example

In example 6.48 we show that, depending on the particular bipartition of the universe Ω into I and D, we may end up with different composites.

example 6.48

We consider components Γ_5 and Δ_5, see figure 6.25.

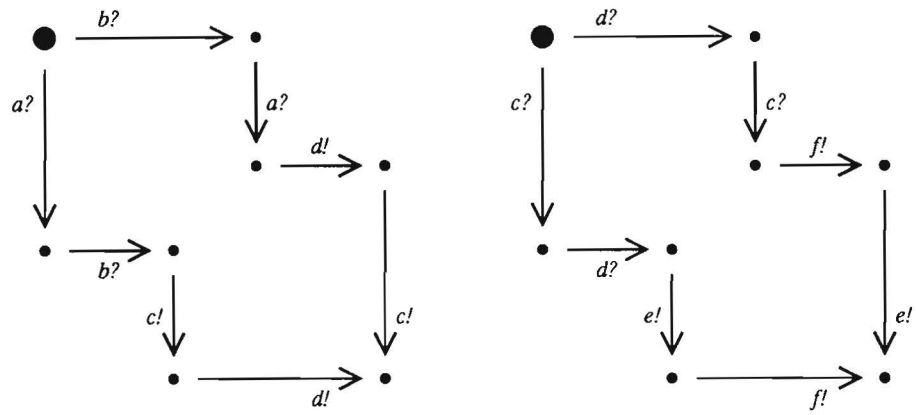

figure 6.25a figure 6.25b

State graphs of components Γ_5 (figure 6.25a) and Δ_5 (figure 6.25b).

We now study the composite of these two components in the four different ways to connect them.

- Let Γ_5 and Δ_5 have a direct connection. We find that $\mathbf{t}(\mathbf{ptr}(\Gamma_5\ \boldsymbol{COMPNCIH}_{\varnothing}\Delta_5)) = \mathbf{pref}\{abef, bafe\}$.

- Let Γ_5 and Δ_5 have a mixed connection such that the commports to which c is associated are indirectly connected and the commports to which d is associated are directly connected. We find that $\mathbf{t}(\mathbf{ptr}(\Gamma_5\ \boldsymbol{COMPNCIH}_{\{c\}}\Delta_5)) = \mathbf{pref}\{abef, abfe, bafe\}$.

- Let Γ_5 and Δ_5 have a mixed connection such that the commports to which c is associated are directly connected and the commports to which d is associated are indirectly connected. We find that $\mathbf{t}(\mathbf{ptr}(\Gamma_5\ \boldsymbol{COMPNCIH}_{\{d\}}\Delta_5)) = \mathbf{pref}\{abef, baef, bafe\}$.

- Let Γ_5 and Δ_5 have an indirect connection. We find that $\mathbf{t}(\mathbf{ptr}(\Gamma_5\ \boldsymbol{COMPNCIH}_{\{c,d\}}\Delta_5)) = \mathbf{pref}\{abef, abfe, baef, bafe\}$.

We conclude that the condition for composition, viz. $\Gamma_5\ \boldsymbol{NICIH}_I\Delta_5$ (see definition 6.35, "***COMPNCIH***"), is satisfied in all four cases (for the appropriate alphabet I, of course). The composite depends on the particular alphabet I.

end of example

6.1.5 Interpretation of the composition method

In the beginning of section 6.1 we have stated that the composite of two components is the result of the composition of these components that, under the given correctness concerns, is maximal with respect to both the inputs that are guaranteed to be accepted by it and the outputs that might be produced by it. In this section we have been concerned with only one correctness concern, viz. "absence of computation interference hazard". The maximality of the composite ***COMPNCIH*** under absence of computation interference hazard follows from the way in which we have combined the trace structures of the two components in definition 6.11, "***totcom***", from the subsequent deletion of only those traces that give rise to computation interference hazard in definition 6.24, "***totcomncih***", and from the hiding in definition 6.29, "***extcomncih***", such that no computation interference hazard is present in the resulting composite.

6.2 Composition without transmission interference hazard

In this section we deal – in addition to absence of computation interference hazard – with the correctness concern *absence of transmission interference hazard*.

6.2.0 Transformation into computation interference hazard

We deal with the additional correctness concern absence of transmission interference hazard by transforming it into computation interference hazard, see section 3.3. In order to apply this technique we have to define trace set(s) that model "transmission interference hazard". For this reason we define **tihi**.

definition 6.49 **tihi**

For component Γ and alphabet A, we define trace set $\mathbf{tihi}_A\Gamma$ by:

$$\mathbf{tihi}_A\Gamma \stackrel{\text{def}}{=} \{a,s : a \in (\mathbf{i}\Gamma \cap A) \wedge s \in (\mathbf{a}\Gamma)^* \wedge saa \in \mathbf{t}\Gamma : saa\}$$

end of definition

By $\mathbf{tihi}_{I \cap \mathbf{o}\Delta}\Gamma$, see definition 6.49, we denote the trace set that is associated with transmission interference hazard between the indirectly connected input commports of component Γ and their matching output commports of component Δ; the symbols of $\mathbf{i}\Gamma \cap \mathbf{o}\Delta \cap I$ are associated with these commports.

We transform transmission interference hazard into computation interference hazard by reducing $\mathbf{ptr}\,\Gamma$ to $\mathbf{ptr}(\mathbf{CBNTIHI}_A\,\Gamma)$.

definition 6.50 **CBNTIHI**

For component Γ and alphabet A, component $\mathbf{CBNTIHI}_A\,\Gamma$ is defined by:

$$\mathbf{io}(\mathbf{CBNTIHI}_A\,\Gamma) \stackrel{\text{def}}{=} \mathbf{io}\,\Gamma$$

$$\mathbf{ptr}(\mathbf{CBNTIHI}_A\,\Gamma) \stackrel{\text{def}}{=} \mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\,\Gamma, \mathbf{tihi}_A\,\Gamma)$$

end of definition

6.2.1 Condition for composition

From definition 6.22, "***NICIH***", we infer condition $\Gamma\,\boldsymbol{NICTIH}_I\,\Delta$ for the definition of $\Gamma\,\boldsymbol{COMPNCTIH}_I\,\Delta$.

definition 6.51 ***NICTIH***

For i/o-connectable components Γ and Δ and alphabet I, we define predicate $\Gamma\,\boldsymbol{NICTIH}_I\,\Delta$ by:

$$\Gamma\,\boldsymbol{NICTIH}_I\,\Delta \stackrel{\text{def}}{=} (\mathbf{CBNTIHI}_{I\cap \mathbf{o}\Delta}\,\Gamma)\,\boldsymbol{NICIH}_I\,(\mathbf{CBNTIHI}_{I\cap \mathbf{o}\Gamma}\,\Delta)$$

end of definition

The condition $\Gamma\,\boldsymbol{NICTIH}_I\,\Delta$ is sufficient on account of definition 6.22. Furthermore, if i/o-connectable components Γ and Δ can be connected under alphabet I with *no initial computation and no initial transmission interference hazard*, it has to be satisfied.

From property 6.23, "***NICIH*** is symmetric", we infer the symmetry of ***NICTIH***.

property 6.52 ***NICTIH is symmetric***

For i/o-connectable components Γ and Δ and alphabet I,

$$\Gamma\,\boldsymbol{NICTIH}_I\,\Delta = \Delta\,\boldsymbol{NICTIH}_I\,\Gamma$$

end of property

6.2.2 Composite of two components

The component, that is *the composite of* Γ *and* Δ *under* I *without computation interference hazard and without transmission interference hazard*, is denoted by $\Gamma\, \textit{COMPNCTIH}_I\, \Delta$.

definition 6.53 ***COMPNCTIH***

For i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\, \textit{NICTIH}_I\, \Delta$, component $\Gamma\, \textit{COMPNCTIH}_I\, \Delta$ is defined by:

$$\Gamma\, \textit{COMPNCTIH}_I\, \Delta \stackrel{\text{def}}{=} (\mathrm{CBNTIHI}_{I \cap o\Delta}\, \Gamma)\, \textit{COMPNCIH}_I\, (\mathrm{CBNTIHI}_{I \cap o\Gamma}\, \Delta)$$

end of definition

From property 6.36, "***COMPNCIH*** is symmetric", property 6.52, "***NICTIH*** is symmetric", and definition 6.53, "***COMPNCTIH***", we infer the symmetry of ***COMPNCTIH***.

property 6.54 ***COMPNCTIH*** *is symmetric*

For i/o-connectable components Γ and Δ and alphabet I such that $\Gamma\, \textit{NICTIH}_I\, \Delta$,

$$\Gamma\, \textit{COMPNCTIH}_I\, \Delta = \Delta\, \textit{COMPNCTIH}_I\, \Gamma$$

end of property

From property 6.37, "***COMPNCIH*** is associative", definition 6.51, "***NICTIH***", and definition 6.53, "***COMPNCTIH***", we infer property 6.55.

property 6.55 ***COMPNCTIH*** *is associative*

For alphabet I and components Γ, Δ, and Θ such that each pair of them is i/o-connectable,

(i) $(\Gamma\, \textit{COMPNCTIH}_I\, \Delta)\, \textit{NICTIH}_I\, \Theta = \Gamma\, \textit{NICTIH}_I\, (\Delta\, \textit{COMPNCTIH}_I\, \Theta)$

(ii) $(\Gamma\, \textit{COMPNCTIH}_I\, \Delta)\, \textit{COMPNCTIH}_I\, \Theta = \Gamma\, \textit{COMPNCTIH}_I\, (\Delta\, \textit{COMPNCTIH}_I\, \Theta)$

end of property

As in property 6.37, in property 6.55 (i) either the left and right hand side both hold or each of them "either is not defined or does not hold". In property 6.55 (ii) either the left and right hand side both are defined or neither is defined; if both are defined then they are equal.

In example 6.56 we illustrate composition without transmission interference hazard.

example 6.56

We consider components Γ_6 and Δ_6, see figure 6.26.

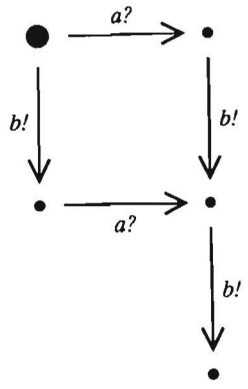

figure 6.26a

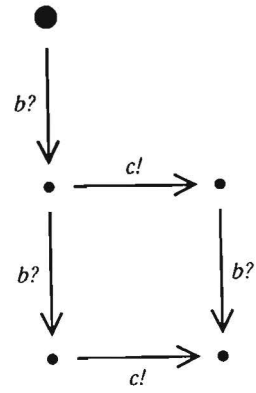

figure 6.26b

State graphs of components Γ_6 (figure 6.26a) and Δ_6 (figure 6.26b).

We are interested in component Γ_6 ***COMPNCTIH***$_{\{b\}}\Delta_6$. From definition 6.53, "***COMPNCTIH***", we infer that we have to calculate **CBNTIHI**$_{\{b\}\cap o\Delta}\Gamma_6$ and **CBNTIHI**$_{\{b\}\cap o\Gamma}\Delta_6$. Since $b\notin o\Delta$ and $b\in o\Gamma$, this amounts to computing **CBNTIHI**$_{\varnothing}\Gamma_6$ and **CBNTIHI**$_{\{b\}}\Delta_6$. From definition 6.50, "**CBNTIHI**", we infer that we have to calculate **tihi**$_{\varnothing}\Gamma_6$ and **tihi**$_{\{b\}}\Delta_6$. From definition 6.49, "**tihi**", we conclude that **tihi**$_{\varnothing}\Gamma_6=\varnothing$ and **tihi**$_{\{b\}}\Delta_6=\{bb\}$. Since Γ_6 ***NICTIH*** Δ_6, cf. definition 6.51, "***NICTIH***", we can calculate component Γ_6 ***COMPNCTIH***$_{\{b\}}\Delta_6$, see figure 6.27a.

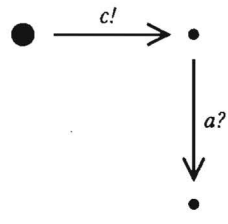

figure 6.27a

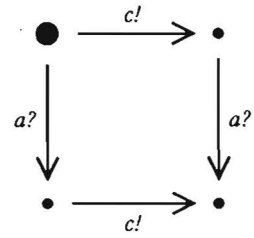

figure 6.27b

State graphs of components Γ_6 ***COMPNCTIH***$_{\{b\}}\Delta_6$ (6.27a) and Γ_6 ***COMPNCIH***$_{\{b\}}\Delta_6$ (6.27b).

To show the difference with the composite when absence of transmission interference is not a correctness concern, we show the state graph of component Γ_6 ***COMPNCIH***$_{\{b\}}\Delta_6$ in figure 6.27b.

end of example

In property 6.57 we present the unity element of the composition operator ***COMPNCTIH***.

property 6.57 *unity element of* $COMPNCTIH_I$

Given is component Γ. Let component Δ be such that $\mathbf{a}\Delta=\varnothing$ and $\mathbf{t}(\mathbf{ptr}\,\Delta)=\{\varepsilon\}$.

(i) Γ and Δ are i/o-connectable

(ii) for alphabet I, $\Gamma\, NICTIH_I\, \Delta$

(iii) for alphabet I, $\Gamma\, COMPNCTIH_I \Delta = \Gamma$

end of property

The unity element of $COMPNCTIH_I$ is equal to the unity element of $COMPNCIH_I$, cf. property 6.41. As a consequence, it does not depend on alphabet I.

Analogously to property 6.45 we find property 6.58.

property 6.58

For i/o-connectable components Γ and Δ and alphabet I such that $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \subseteq I$ and $\Gamma\, NICTIH_I\, \Delta$,

$$\mathbf{DIE}(\Gamma\, COMPNCTIH_I\, \Delta) = (\mathbf{DIE}\,\Gamma)\, COMPNCTIH_I (\mathbf{DIE}\,\Delta)$$

end of property

Analogously to property 6.46 we find property 6.59.

property 6.59

For i/o-connectable components Γ and Δ and alphabet I such that $\mathbf{a}\Gamma \cap \mathbf{a}\Delta \subseteq I$ and $\Gamma\, NICTIH_I\, \Delta$,

$$\mathbf{CBDI}(\Gamma\, COMPNCTIH_I\, \Delta) = (\mathbf{CBDI}\,\Gamma)\, COMPNCTIH_I (\mathbf{CBDI}\,\Delta)$$

end of property

6.3 Decomposition

When we discuss decomposition we are motivated by concerns about the implementation of specifications. The decomposition problem that we address is known as *factorization*, cf. [Fang87]: in this technique a specification Γ_S and a (desired) part Γ_M of a solution of this specification are given; the problem amounts to calculating the specification Γ_X of the remainder, whenever under the given correctness concerns such a remainder exists. Of course, the task of calculating Γ_X has to be accomplished under the given correctness concerns. In this monograph we are concerned with the correctness concerns "absence of computation interference hazard" and "absence of transmission interference hazard". Both correctness concerns are symmetric w.r.t. the specification (Γ_S) and all parts of the solution. Due to this symmetry factorization is equal to composition. We notice that factorization is concerned with the closed composition of three parts, see figure 6.28.

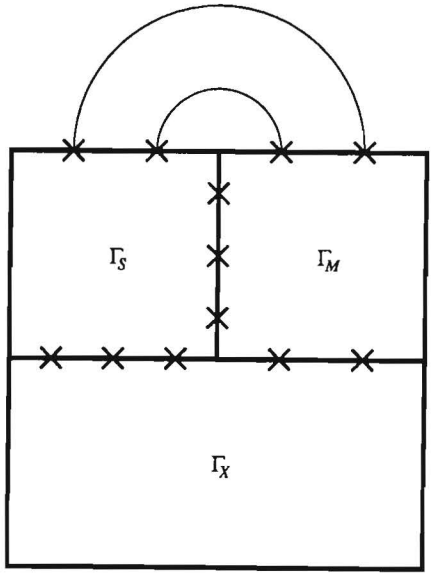

figure 6.28
Factorization of Γ_S into Γ_M and Γ_X .

When factorizing Γ_S into Γ_M and Γ_X , we deal with a mixed connection between Γ_S and Γ_M . The connection between on the one hand the composite of Γ_S and Γ_M and on the other hand Γ_X is direct; a possible indirect or mixed connection between this composite and Γ_X is left to the next step(s) in the factorization.

In our Communication Model we calculate the specification of the remainder mentioned above by composing the Γ_S with Γ_M. The specification Γ_X of the remainder is the reflection of this composite. As mentioned in section 6.1 this composite is maximal w.r.t. outputs that might be produced and maximal w.r.t. inputs that are guaranteed to be accepted. Since absence of computation interference hazard is a correctness concern the specification of the remainder, which is the reflection of the composite calculated in this way, is maximal w.r.t. the inputs that is has to accept and maximal w.r.t. the outputs that it might produce.

The following examples have been shown by Ebergen, cf. [Ebergen 87]. In example 6.60 we show the decomposition of a wire component into two wire components. In spite of our notational convention we will use Γ, Δ, and Θ to denote components in these examples.

example 6.60

We consider components Γ_7, Δ_7, and Θ_7; all three model wire elements, cf. example 2.47; they are given by:

$$\begin{array}{lll} \mathbf{o}\Gamma_7 \stackrel{\text{def}}{=} \{b\}, & \mathbf{i}\Gamma_7 \stackrel{\text{def}}{=} \{a\}, & \mathbf{t}(\mathbf{ptr}\,\Gamma_7) \stackrel{\text{def}}{=} \mathbf{pref}\{ab\}, \\ \mathbf{o}\Delta_7 \stackrel{\text{def}}{=} \{c\}, & \mathbf{i}\Delta_7 \stackrel{\text{def}}{=} \{a\}, & \mathbf{t}(\mathbf{ptr}\,\Delta_7) \stackrel{\text{def}}{=} \mathbf{pref}\{ac\}, \\ \mathbf{o}\Theta_7 \stackrel{\text{def}}{=} \{b\}, & \mathbf{i}\Theta_7 \stackrel{\text{def}}{=} \{c\}, & \mathbf{t}(\mathbf{ptr}\,\Theta_7) \stackrel{\text{def}}{=} \mathbf{pref}\{cb\}. \end{array}$$

Since $\Gamma_7 = (\Delta_7\ \boldsymbol{COMPNCIH}_{\varnothing}\ \Theta_7)$, we conclude that Γ_7 can be decomposed into Δ_7 and Θ_7 such that the connection of Δ_7 and Θ_7 is direct and there is absence of computation interference hazard, see figure 6.29.

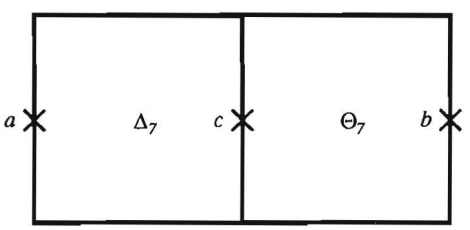

figure 6.29
Decomposition of wire component into two wire components.

Notice that also $\Gamma_7 = (\Delta_7\ \boldsymbol{COMPNCIH}_{\{c\}}\ \Theta_7)$, and that $\Gamma_7 = (\Delta_7\ \boldsymbol{COMPNCTIH}_{\{c\}}\ \Theta_7)$.

end of example

In example 6.61 we show the decomposition of a wire component that models a wire element into components that model a fork element and a Muller-C element.

example 6.61

We consider components Γ_8, Δ_8, and Θ_8; Γ_8 models a wire element, see example 2.47, Δ_8 models a fork element, see example 2.49, and Θ_8 models a Muller-C element, see example 2.48; they are given by:

$$\begin{array}{lll} \mathbf{o}\Gamma_8 \stackrel{\text{def}}{=} \{b\}, & \mathbf{i}\Gamma_8 \stackrel{\text{def}}{=} \{a\}, & \mathbf{t}(\mathbf{ptr}\,\Gamma_8) \stackrel{\text{def}}{=} \mathbf{pref}(\{ab\})^*, \\ \mathbf{o}\Delta_8 \stackrel{\text{def}}{=} \{c,d\}, & \mathbf{i}\Delta_8 \stackrel{\text{def}}{=} \{a\}, & \mathbf{t}(\mathbf{ptr}\,\Delta_8) \stackrel{\text{def}}{=} \mathbf{pref}(\{acd,adc\})^*, \\ \mathbf{o}\Theta_8 \stackrel{\text{def}}{=} \{b\}, & \mathbf{i}\Theta_8 \stackrel{\text{def}}{=} \{c,d\}, & \mathbf{t}(\mathbf{ptr}\,\Theta_8) \stackrel{\text{def}}{=} \mathbf{pref}(\{cdb,dcb\})^*. \end{array}$$

Since $\Gamma_8 = (\Delta_8\ \boldsymbol{COMPNCIH}_{\varnothing}\ \Theta_8)$, we conclude that Γ_8 can be decomposed into Δ_8 and Θ_8 such that the connection of Δ_8 and Θ_8 is direct and there is absence of computation interference hazard, see figure 6.30.

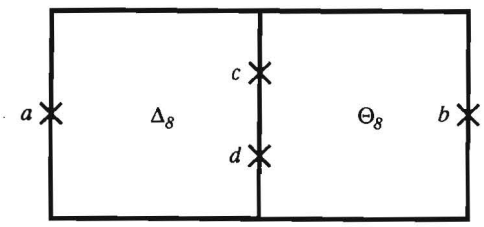

figure 6.30
Decomposition of wire component into fork component and Muller-C component.

Notice that also $\Gamma_8 = (\Delta_8\ \boldsymbol{COMPNCIH}_{\{c\}}\ \Theta_8)$, and that $\Gamma_8 = (\Delta_8\ \boldsymbol{COMPNCTIH}_{\{c\}}\ \Theta_8)$.
end of example

6.4 Other correctness concerns

The method for constructing the composite of components presented in this chapter is also suited to deal with other correctness concerns. E.g. we can deal with "absence of ambiguous quiescence hazard", cf. subsection 3.3.1.0, instead of or in addition to "absence of transmission interference hazard".

7

Concluding remarks

In chapter 2 we have introduced our Communication Model as a formal abstraction of 'the underlying physics'. In this model we distinguish direct, indirect, and mixed connections of components. Furthermore, we deal with interpretational issues like inputs and outputs in this model; by doing so, we do not saddle the trace theory formalism with this burden. Trace theory is a formalism that is used in several ways in our Communication Model. Furthermore, we believe that the presence of a Communication Model has enabled us to pinpoint the abstraction from module to component, see subsection 2.2.3. We also carefully distinguish between the communication behavior of components and the communication of a channel between them.

In section 3.3 we presented a technique that transforms "undesired phenomenon hazards" into "computation interference hazard". We showed some applications of this technique in the subsequent chapters. The example of the application of this technique in which we deal with the correctness concern "absence of ambiguous quiescence hazard", see subsection 3.3.1, indicates that many correctness concerns (even some liveness properties) can be incorporated in our Communication Model in this way. This transformation technique is also the basis for the composition operators defined in chapter 6, where it suggests a way to define new composition operators that include other correctness concerns. The definitions of our composition operators for mixed connections of components are helpful tools to compare and combine synchronous and asynchronous design methods.

We have formally defined absence of computation interference hazard in chapters 3 and 4 (for direct and indirect connections of components, respectively). Absence of computation interference hazard is the basic correctness concern in this monograph. The distinction between the reception and the acceptance of a signal provides the context that is needed for the discussion of computation interference hazard.

In chapter 4 we have addressed delay-safety. We do not talk about 'delay-safe circuits': delay-safety is not a property of a physical circuit. At the circuit level delay-safety is just an assumption, viz. the value of the delay of a signal that is sent from one terminal via a wire to another terminal is nonnegative. Of course, we could try to define the predicate "delay-safe" for circuits; this would amount to something like: "the correctness of the functioning of the circuit does not depend on the values of the delays in the wires of the circuit". Notice that the functioning of the circuit may depend on the values of these delays: e.g., depending on the values of the delays the circuit may behave in a different –but correct !– way. In order to define this predicate "delay-safe", however, one does not only need a circuit, but also a description of the correctness of its functioning and a method to check whether this correctness does or does not depend on the values of the delays in the wires of the circuit.

In chapters 4 and 5 we present theorems that link in our Communication Model the constructive definitions of trace structures **dse**, **cbds**, **die**, and **cbdi**, to the intuitive definitions of components **DSE**, **CBDS**, **DIE**, and **CBDI**, respectively. These are tools that help a designer to decide whether he wants to use delay-safe (or delay-insensitive) communication or not, since they can be used to indicate the limitations of delay-safe and delay-insensitive communication, see subsection 4.2.5 and subsection 5.1.1, respectively. In these subsections we address so-called 'off-the-shelf' mechanisms, cf. [Molnar85]:

In the context of delay-safe communication we present in chapter 5 an intuitive definition of "absence of transmission interference hazard". We show furthermore, that "delay-insensitive communication" is equal to "delay-safe communication without transmission interference hazard".

In chapter 6 we address composition. There we deal with the general case: mixed connections of components. We generalize composability to "general composability"; we also present a generalization of "composability diagrams", viz. "general composability diagrams". General composability diagrams can be used to check readily whether two traces are generally composable under some given iobip or not. In this chapter we present necessary and sufficient conditions for composition in two cases: (i) under the correctness concern absence of

computation interference hazard, and (ii) under the correctness concerns absence of computation interference hazard and absence of transmission interference hazard. Furthermore, we address factorization in this chapter. Factorization is the decomposition problem, in which the specification and a part of the desired solution are given and the remainder has to be calculated. Factorization is equal to composition if and only if all correctness concerns are symmetric w.r.t. the specification and all parts of the solution.

7.0 Formal definitions of delay-insensitive

In this section we present some links between the pieces of research that have been carried out within the field "delay-insensitivity". Furthermore, we show relations between our work and the work of other researchers.

7.0.0 Relation between self-timed and delay-insensitive

The class of self-timed circuits has been introduced by Seitz, see [Seitz80]. He distinguishes time geometry, i.e. time metric, and time topology, i.e. a partial order on the occurrences of events. The relation between the time metric and the time topology is inside the self-timed elements. Self-timed elements either are synchronous systems with an internal clock that can be stopped synchronously and restarted asynchronously or they are speed-independent circuits. The design of self-timed circuits has two principal facets: the design of elements and the design of systems of interconnected elements. Along the seam between those subjects are conventions for self-timed signaling. Equipotential regions have been introduced in order to try to assure consistent physical meaning for the relations that hold within them; it is necessary that a self-timed element is contained in at least one equipotential region; the set of equipotential regions covers all of the elements of the self-timed system.

Seitz argues that a strict protocol of signaling conventions has to be imposed throughout the system in order to deal with the complexity of the design, see [Seitz80]. Two-phase handshaking and four-phase handshaking are such protocols. Van de Snepscheut has given a theoretical foundation, see [Van de Snepscheut85]. He defines the "agglutinate", which really is the same operator as the composition operator "©" which we presented in [Schols85]; the only difference is that van de Snepscheut was concerned with the external communication behavior of the composition, whereas we were interested in the internal communication. Van de Snepscheut detects computation interference hazard. His delay-insensitive communication is more restrictive than ours; our composition operator ***COMPNCIH***, see chapter 6, is a generalization of his agglutinate. Building on van de Snepscheut's foundation, Martin shows how a compilation can be performed from a specification to a self-timed circuit in which four-phase handshaking is used for the communication between elements that are not in the same equipotential region, see [Martin85b, Martin86, Martin87]. Among the most significant results of Martin's group is the design of an asynchronous microprocessor, see [Martin – Burns – Lee – Borkovic – Hazewindus89]. The specification language from which Martin starts his compilation is CSP extended with the communication primitive "probe", see [Martin85a]. A detailed overview of Martin's method is given in [Martin90].

We have suggested an alternative approach to Martin's method using invariants. This has been formalized by Langenberg, see [Langenberg92]. Langenberg also addresses *overspecification* in this context. De Graaff has suggested a design method that is somewhat similar to Martin's method, see [de Graaff86]. De Graaff introduces the distinction between the acceptance/reception of a signal by a mechanism and the 'observation' of that signal by this mechanism. Based upon Martin's approach, van Berkel has developed a decomposition method that leads to 'delay-insensitive circuits', see [van Berkel92]. In the graduate student project VOC at Eindhoven University of Technology methods for designing 'delay-insensitive circuits' are investigated and developed, see [Bisseling – Eemers – Kamps – Peeters90]; the ultimate goal is to build a silicon compiler for translating parallel computations into 'delay-insensitive VLSI circuits'.

7.0.1 Modular approach to delay-insensitivity

Keller, see [Keller74], defines a "delay-insensitive network" as follows:

> A network is called *delay-insensitive* if its external behavior remains unchanged, regardless of whether any number of delay elements are inserted into, or removed from any lines.

In the approach in this monograph we do not require the external behavior to remain unchanged, regardless of the amount of delay in such lines. We allow the external behavior to change, as long as it remains correct w.r.t. its specification; here, the environment plays an important role, see section 6.0. Furthermore, Keller's definition suggests that the delay along such a line is fixed, although unknown; in the approaches mentioned below, the delay constraint has been strengthened to allow values of the delays in a given line to be distinct. Keller's definition refers to delays in "any lines". When such a constraint is imposed rigorously on all parts of the circuit, it results in a very restricted class of delay-insensitive networks, see [Seger88]. This constraint is weakened in the approaches mentioned below to delays in the lines that connect so-called modules: lines inside these modules are not being considered for inserting such delay elements.

Molnar has introduced the "Foam Rubber Wrapper metaphor", see [Molnar–Fang–Rosenberger85]. With respect to the communication in the channel, it assumes that the values of the delays of the commsigs are nonnegative. With respect to the communication behavior of components, it assumes that there is no computation interference hazard, see [Schols88]. Based upon this intuitive notion three formalizations of delay-insensitivity arose, see [Udding84, Schols85, Black86]. Furthermore, this notion inspired Verhoeff, Ebergen, and Dill to define delay-insensitivity formally, see [Verhoeff85, Ebergen87, Dill88].

Udding has classified 'delay-insensitive circuits', see [Udding84]. The smallest class is called the "synchronization class". No data communication is possible in this class, since no choice can be made: the only way to disable a communication action is to let it take place. The second class is called the "data communication class". In this class choice between inputs is allowed; this enables data communication. The communication in this class depends on the role of inputs and outputs; interchanging inputs and outputs yields, in general, a circuit that is not in the data communication class. The 'regular circuits' in this class are considered to be synthesizable. Next comes the "arbitration class". This class contains nondeterministic behavior: in addition to choice between inputs, choice between outputs is allowed in this class. The greatest class is called the "delay-insensitive class"; it is also called C_4. The synchronization class is a subset of the data communication class. The latter is a subset of the arbitration class which, in turn, is a subset of the delay-insensitive class. Except for the arbitration class, all classes are closed under composition, see [Verhoeff85]. Udding's classes cannot be used to classify the communication behaviors of components that communicate delay-insensitively: such a communication behavior need not belong to the delay-insensitive class, see **CBDI** in subsection 5.1.0.3. These classes, except for the data communication class, can be used to classify the communication in delay-insensitive channels, cf. section 6.0. The data communication class is not suited to this purpose, since it has been defined asymmetrically w.r.t. the two parts of the alphbip; these two parts are called input and output in [Udding84]. Udding's classes can be used to classify components, say Γ, according to the communication in the channel between Γ and its maximal partner, when they communicate delay-insensitively, see subsection 5.1.0.1. Udding's classes can be interpreted at the boundary between **DIE** Γ and **DIE** Δ in figure 7.0.

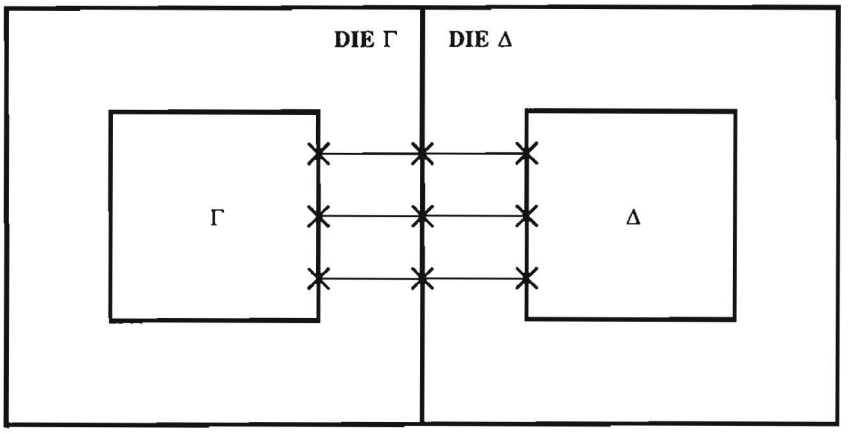

figure 7.0
Components Γ and Δ and their delay-insensitive enclosures.

We have eliminated the correctness concern "absence of transmission interference hazard" from the conditions imposed on Udding's delay-insensitive class, see [Schols85]. Verhoeff has eliminated this correctness concern from the conditions imposed on all four classes, see [Verhoeff85]; this yields the classes $\mathbf{D_1}$ through $\mathbf{D_4}$ ($\mathbf{D_4}$, see subsection 4.1.0, being the largest of the four). These classes can be interpreted at the boundary between **DSE** Γ and **DSE** Δ in figure 7.1, see subsection 5.2.3.

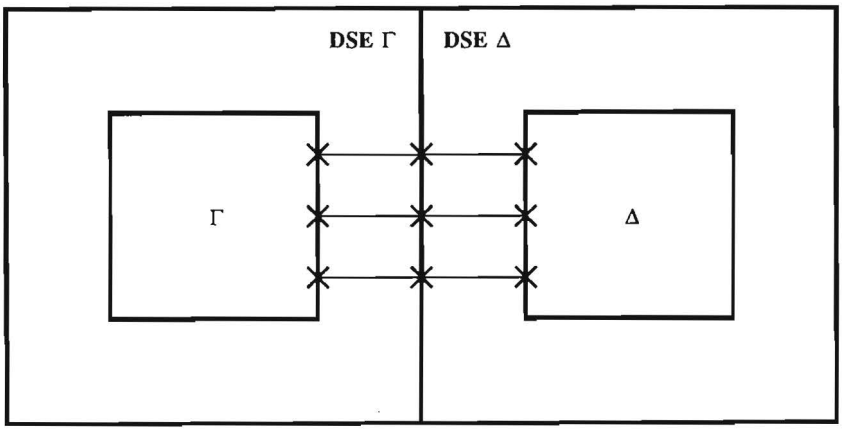

figure 7.1
Components Γ and Δ and their delay-safe enclosures.

Verhoeff has shown which protocols are suited for delay-insensitive data communication, see [Verhoeff88].

7.0.2 Delay-safety and delay-insensitivity

We have previously defined a composition operator that can be used to check delay-safety and also to calculate the smallest delay-safe communication that includes the original communication, see [Schols85]. We have proven that delay-safety can be separated from "absence of transmission interference hazard", see [Schols85, Verhoeff-Schols85]. We show this separation of concerns in [Schols88]. "Delay-safe communication without transmission interference hazard" is called "delay-insensitive communication", see chapter 5. In this monograph we apply our earlier results, see [Schols85], to communication in channels; we discuss the impact of these results on the communication behaviors of components. Furthermore, we address composition in this monograph.

Black uses infinite trace theory, i.e. he allows for traces of infinite length, see [Black86]. He extends our earlier definition of delay-safety using infinite trace theory. Furthermore, he deals explicitly with the 'capacity' of the "links", which form the connection between indirectly connected commports.

Verhoeff has introduced a Delay-demon, that models the non-negativity of the delays in the "links" in the channels, see [Verhoeff85]. Like Udding, Verhoeff is concerned with the communication behavior of components; he also distinguishes four classes. He does not, however, include "absence of transmission interference hazard" in the definitions of his classes. He deals separately with the 'capacity' of the "links" in the channels.

Ebergen has defined Wire components, see [Ebergen87]. He does not distinguish the "links" in the channels from the other components; in his approach transmission interference hazard is a special case of computation interference hazard, viz. at the inputs to the Wire components. Ebergen defines his Wire components in such a way that they have a 'capacity' of one commsig. Furthermore, Ebergen has defined a decomposition method for the translation of specifications (in particular: programs) into 'delay-insensitive circuits', see [Ebergen87]. [Ebergen88] is a good introduction to Ebergen's method. The decomposition problem has also been attacked by Fang; he addresses "factorization" in [Fang87], see also section 6.3.

Dill defines delay-safety for CSP-like processes. His processes are quadruples: input alphabet, output alphabet, set of successful traces, set of failure traces. The set of successful traces and the set of failure traces need not be disjunct.

Josephs and Udding have developed an "algebra for delay-insensitive circuits", see [Josephs – Udding 89] and [Josephs – Udding 90]. Their approach is based upon CSP. Their formalism is such that every specification that is syntactically correct is 'delay-insensitive'. On the one hand this is convenient when one wants to end up with a 'delay-insensitive specification'; on the other hand, it is difficult to express in their formalism the functionality that one desires. Of course, arguing about delay-insensitivity is not possible (nor necessary) within their formal framework.

If we disregard transmission interference hazard, all of the above formalizations are equivalent. Although the definitions of the formalizations differ very much in form, none makes it easy to prove by hand that a particular communication is delay-insensitive. They can more easily be used to show that a particular communication is not delay-insensitive: find a case and show that it does not satisfy the requirements for delay-insensitivity.

7.0.3 Fairness and delay-insensitivity

In the past much discussion has gone on concerning the question whether delay-insensitive fair arbitration is possible or not. People interested in building arbiters are referred to [Chaney 86] and [Unger 80]. Martin has shown that delay-insensitive fair arbiters can be built, see [Martin 85b, Dill 88]; in his design the communication that is internal to the fair arbiter is delay-insensitive. On the other hand it has been argued that delay-insensitive fair arbitration is not possible, see [Udding 85, Moll 85, Cox 85]. Our conclusion is: we are able to build fair arbiters that internally communicate delay-insensitively, but when an arbiter communicates delay-insensitively with its environment this arbiter may not be fair any more, seen from the point of view of this environment. The lesson to be learned from this is: if we need a fair arbiter, we can build it such that the internal communication is delay-insensitive; we have to take care that the communication between this arbiter and its environment is not delay-insensitive.

7.0.4 Testing for delay-insensitivity

Burstyn and Udding have written a program to test automatically whether the composition of a number of delay-insensitive modules is correct in the sense that no possible sequence of communication actions can result in computation interference hazard. This program can be used to verify that a particular communication is delay-insensitive, see [Burstyn86]. The program accomplishes this by, first, internally generating the reflection of the given component and Ebergen's Wire components. The Wire components are used to connect the component to its reflection. Next, the program tests the resulting composition for computation interference hazard. If this test is negative, the trace structure of the original component is in Udding's delay-insensitive class, i.e. $\mathbf{C}_4$.

7.1 Topics for further research

In chapter 2 we remarked that it is possible to infer a comminstorder of a module from the causal ordering of signals exchanged by a mechanism. Some initial exercises showed that it might be interesting to consider Dynamical Systems theory as the underlying physical model; it seems promising to pursue a formal relation between Dynamical Systems theory and our Communication Model.

We presented abstractions in chapter 2: a component is an equivalence class of modules, and a channel is an equivalence class of interconnections. Delay-safety and delay-insensitivity could be modeled using modules and interconnections instead of components and channels. In this case, communication behaviors would be sets of trace sets rather than trace sets. The outcome of such work will yield interesting information about what is lost by our abstraction; e.g., ambiguous quiescence hazard can be introduced by this abstraction, see remark 3.16.

For the operators that have been defined in this monograph programs can be developed; these programs might serve as a tool for designers. Furthermore, they might be integrated in larger development environments. Van der Heijden and Teunissen have developed a program for the operator **DIE**, which is referred to as "**DECNTIH**" by them, see [van der Heijden – Teunissen 89].

Within our Communication Model, both synchronous and asynchronous communication can be addressed. We integrate them in chapter 6: the composition operators in this chapter deal with mixed connections of components. As such, these operators may constitute a step towards the integration of design techniques based on synchronous and asynchronous communication models, see section 0.0. We believe that in the next decade synchronous and asynchronous design techniques will end to be competitors: they will be integrated in large development environments in which they both can be used by a designer, depending on the particular design task.

In this monograph we have been concerned with the limitations of delay-safe and delay-insensitive communication. Often a strict protocol of signaling conventions is imposed throughout a system in order to deal with the complexity of the design, cf. [Seitz80]. The operators presented in this monograph can be used to check whether such a methodical approach is consistent with delay-safe and delay-insensitive communication.

Appendix A

Proofs

This appendix has three sections. Section A.3 contains the proof of the theorem of chapter 3. Sections A.4 and A.5 contain the proofs of the lemmas and theorems of chapters 4 and 5, respectively.

A.3 Computation interference hazard

theorem 3.14

Let UndesPh be some undesired phenomenon. Let trace set S be associated with UndesPh. Let Γ be a component such that $(\mathbf{A}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \cap S : l(s \restriction \mathbf{i}\Gamma) > 0)$. --- We define component Γ' by $\Gamma' \stackrel{\text{def}}{=} <\mathbf{io}\,\Gamma, \mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\,\Gamma, S)>$.

Then Γ' is the maximal (w.r.t. trace structure inclusion) component such that

(i) $\mathbf{io}\,\Gamma' = \mathbf{io}\,\Gamma$,

(ii) $\mathbf{ptr}\,\Gamma' \subseteq \mathbf{ptr}\,\Gamma$,

(iii) Γ' has absence of UndesPh hazard.

proof

Let UndesPh be an undesired phenomenon hazard. Let trace set S be associated with UndesPh. Let Γ be a component. Let component Γ' be defined by $\Gamma' \stackrel{\text{def}}{=} <\mathbf{io}\,\Gamma, \mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\,\Gamma, S)>$.

From the definition of Γ' follows (i). From definition 1.34, "**redts**", follows (ii). Since all traces of $\mathbf{t}(\mathbf{ptr}\,\Gamma) \cap S$ are missing in $\mathbf{t}(\mathbf{ptr}\,\Gamma')$, cf. definition 1.34, "**redts**", we infer that (iii) holds.

In order to argue the maximality of Γ', we consider a trace t such that $t \in (\mathbf{t}(\mathbf{ptr}\,\Gamma) \setminus \mathbf{t}(\mathbf{ptr}\,\Gamma'))$. Using $\mathbf{ptr}\,\Gamma' = \mathbf{redts}(\mathbf{ptr}\,\Gamma, \mathbf{i}\,\Gamma, S)$ we infer from property 1.36, that $(\mathbf{E}\, x, a : x \in (\mathbf{a}\,\Gamma)^* \wedge a \in \mathbf{i}\,\Gamma \wedge xa\,\mathbf{prefix}\, t : x \in \mathbf{t}(\mathbf{ptr}\,\Gamma') \wedge xa \notin \mathbf{t}(\mathbf{ptr}\,\Gamma'))$. From definition 1.34, "**redts**", we infer that

$$(\mathbf{E}\, x, y, a : x \in (\mathbf{a}\,\Gamma)^* \wedge y \in (\mathbf{o}\,\Gamma)^* \wedge a \in \mathbf{i}\,\Gamma \wedge xa\,\mathbf{prefix}\, t : x \in \mathbf{t}(\mathbf{ptr}\,\Gamma') \wedge xa \notin \mathbf{t}(\mathbf{ptr}\,\Gamma') \wedge xay \in (\mathbf{t}(\mathbf{ptr}\,\Gamma) \cap S))$$

Given such traces x and y and such a symbol a. The addition of trace t to $\mathbf{t}(\mathbf{ptr}\,\Gamma')$ leads to the presence of xa in $\mathbf{t}(\mathbf{ptr}\,\Gamma')$. Since components cannot be prevented from producing their output comminsts, trace xay should be present, too. Since $xay \in (\mathbf{t}(\mathbf{ptr}\,\Gamma) \cap S)$ and S is associated with UndesPh, we infer that the addition of a trace to $\mathbf{t}(\mathbf{ptr}\,\Gamma')$ introduces UndesPh hazard. We conclude that Γ' is maximal.

end of theorem

A.4 Communicating delay-safely

In this section we present the proofs of the lemmas and theorems of chapter 4.

lemma 4.43

For component Γ,

$$(\mathbf{A}\, t : t \in \mathbf{t}(\mathbf{dse}\,\Gamma) : (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\, t))$$

proof

Given component Γ. Let t be such that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$. We prove this lemma by induction on the length of t.

induction hypothesis

$$(\mathbf{A}\, u : u \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge lu < lt : (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\, u))$$

base: $lt = 0$

$$\begin{array}{ll} & \mathbf{true} \\ = & \{ \text{ property 4.5(i) } \} \\ & \varepsilon\mathbf{c}_{\mathbf{io}\,\Gamma}\,\varepsilon \\ = & \{ \text{ } t = \varepsilon, \text{ since } lt = 0 \text{ } \} \\ & \varepsilon\mathbf{c}_{\mathbf{io}\,\Gamma}\, t \\ \Rightarrow & \{ \text{ property 2.34(i) } \} \\ & (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\, t) \end{array}$$

step: $lt > 0$

Let $t = xa$ for trace x and symbol a; hence, $lx < lt$. From $t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ follows that $a \in \mathbf{a}\Gamma$ and $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\Gamma$

$$\begin{array}{ll} & \mathbf{true} \\ = & \{ \text{ } xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \text{ } \} \\ & xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\ \Rightarrow & \{ \text{ definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{o}\Gamma \text{ } \} \\ & (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge s\mathbf{c}_{\mathbf{io}\,\Gamma}\, x : \#_a s > \#_a x) \\ = & \{ \text{ property 4.11(i), using } a \in \mathbf{o}\Gamma \text{ } \} \\ & (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\, xa) \\ = & \{ \text{ } t = xa \text{ } \} \\ & (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\, t) \end{array}$$

case 1: $a \in \mathbf{i}\Gamma$

$$
\begin{array}{ll}
 & \text{true} \\
= & \quad \{\ xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)\ \} \\
 & xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\
\Rightarrow & \quad \{\ \mathbf{dse}\,\Gamma \text{ is prefix-closed}\ \} \\
 & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\
\Rightarrow & \quad \{\ \text{induction hypothesis, using } lx < lt\ \} \\
 & (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{lo}\,\Gamma} x) \\
\Rightarrow & \quad \{\ \text{property 4.11(iv), using } a \in \mathbf{i}\Gamma\ \} \\
 & (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{lo}\,\Gamma} xa) \\
= & \quad \{\ t = xa\ \} \\
 & (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{lo}\,\Gamma} t)
\end{array}
$$

end of lemma

lemma 4.44

For components Γ and Δ such that $\mathbf{io}\Gamma = \mathbf{io}\Delta$ and $\overline{\Delta}$ *NCIHADS* Γ,

$$(\mathbf{A}\, s, t : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t \in \mathbf{t}(\mathbf{ptr}\,\Delta) \setminus \mathbf{t}(\mathbf{dse}\,\Gamma) : \neg(s\, \mathbf{c}_{\mathbf{io}\Gamma}\, t))$$

proof

Given components Γ and Δ such that $\mathbf{io}\Gamma = \mathbf{io}\Delta$ and $\overline{\Delta}$ *NCIHADS* Γ. Let trace t be such that $t \in \mathbf{t}(\mathbf{ptr}\,\Delta) \setminus \mathbf{t}(\mathbf{dse}\,\Gamma)$. Since $\varepsilon \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, $t \neq \varepsilon$. Let trace x and symbol a be such that $x a \,\mathbf{prefix}\, t$, $a \in \mathbf{a}\Gamma$, $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, $xa \in \mathbf{t}(\mathbf{ptr}\,\Delta)$, and $xa \notin \mathbf{t}(\mathbf{dse}\,\Gamma)$. We first prove that $a \in \mathbf{o}\Gamma$.

true
= { definition 4.29, "*NCIHADS*", using $\overline{\Delta}$ *NCIHADS* Γ and $\mathbf{a}\Gamma = \mathbf{a}\Delta$ }
$(\mathbf{A}\, r, s, b : r \in \mathbf{t}(\mathbf{ptr}\,\overline{\Delta}) \wedge s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{o}\overline{\Delta} \wedge r\, \mathbf{c}_{\mathbf{io}\overline{\Delta}}\, s \wedge \#_b r > \#_b s : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$
⇒ { $xa \in \mathbf{t}(\mathbf{ptr}\,\overline{\Delta})$, since $xa \in \mathbf{t}(\mathbf{ptr}\,\Delta)$, see definition 2.36, "reflection of component" }
$(\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{o}\overline{\Delta} \wedge xa\, \mathbf{c}_{\mathbf{io}\overline{\Delta}}\, s \wedge \#_b xa > \#_b s : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$
= { property 4.9, and calculus }
$(\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{o}\overline{\Delta} \wedge s\, \mathbf{c}_{\mathbf{io}\Delta}\, xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$
= { $\mathbf{io}\Delta = \mathbf{io}\Gamma$ and $\mathbf{o}\overline{\Delta} = \mathbf{i}\Gamma$, see definition 2.36, "reflection of component" }
$(\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge s\, \mathbf{c}_{\mathbf{io}\Gamma}\, xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$
⇒ { definition 4.36, "**dse**", using $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ and $xa \notin \mathbf{t}(\mathbf{dse}\,\Gamma)$ }
$a \notin \mathbf{i}\Gamma$
= { $a \in \mathbf{a}\Gamma$ and $\mathbf{o}\Gamma = \mathbf{a}\Gamma \setminus \mathbf{i}\Gamma$, see property 2.33 }
$a \in \mathbf{o}\Gamma$

We have derived $a \in \mathbf{o}\Gamma$.

true
= { definition 4.36, "**dse**", using $a \in \mathbf{o}\Gamma$, $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, and $xa \notin \mathbf{t}(\mathbf{dse}\,\Gamma)$ }
$(\mathbf{A}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge s\, \mathbf{c}_{\mathbf{io}\Gamma}\, x : \#_a s \leq \#_a x)$
= { predicate calculus }
$(\mathbf{A}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : \neg(s\, \mathbf{c}_{\mathbf{io}\Gamma}\, x \wedge \#_a s > \#_a x))$
= { property 4.11(i), using $a \in \mathbf{o}\Gamma$ }
$(\mathbf{A}\, s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : \neg(s\, \mathbf{c}_{\mathbf{io}\Gamma}\, xa))$
⇒ { property 4.7, using $xa \,\mathbf{prefix}\, t$ and property 2.45(ii) }
$(\mathbf{A}\, u : u \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : \neg(u\, \mathbf{c}_{\mathbf{io}\Gamma}\, t))$

end of lemma

theorem 4.45 *delay-safe enclosure*

For component Γ,

$$\mathbf{ptr}(\mathbf{DSE}\,\Gamma) = \mathbf{dse}\,\Gamma$$

proof

Given component Γ. Let component Δ be such that $\mathbf{io}\,\Delta = \mathbf{io}\,\Gamma$ and $\mathbf{ptr}\,\Delta = \mathbf{dse}\,\Gamma$.

From property 4.42 we conclude that $\Gamma\ \mathit{NCIHDS}\ \bar{\Delta}$. From lemma 4.43 we derive that $(\mathbf{A}\,a,t : a \in \mathbf{o}\,\Gamma \wedge ta \in \mathbf{t}(\mathbf{ptr}\,\Delta) : (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,ta))$. The maximality of Δ follows from lemma 4.44.

end of theorem

lemma 4.49

For component Γ,

$$(\mathbf{A}\, t : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge (\mathbf{E}\, y : y \in \mathbf{t}(\mathbf{dse}\,\Gamma) : t\mathbf{c}_{\mathbf{io}\,\Gamma} y) : t \in \mathbf{t}(\mathbf{dse}\,\Gamma))$$

proof

Given component Γ. Let trace t be such that $t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, and $(\mathbf{E}\, y : y \in \mathbf{t}(\mathbf{dse}\,\Gamma) : t\mathbf{c}_{\mathbf{io}\,\Gamma} y)$. We prove this lemma by induction on the length of t.

induction hypothesis

$$(\mathbf{A}\, u : u \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge lu < lt \wedge (\mathbf{E}\, z : z \in \mathbf{t}(\mathbf{dse}\,\Gamma) : u\mathbf{c}_{\mathbf{io}\,\Gamma} z) : u \in \mathbf{t}(\mathbf{dse}\,\Gamma))$$

base: $lt = 0$

true

$=$ { $lt = 0$ }

$t = \varepsilon$

$\Rightarrow$ { $\varepsilon \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, see definition 4.36, "**dse**" }

$t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

step: $lt > 0$

Let $t = xa$ for trace x and symbol a; hence, $lx < lt$, $a \in \mathbf{a}\Gamma$, $xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, and $(\mathbf{E}\, y : y \in \mathbf{t}(\mathbf{dse}\,\Gamma) : xa\mathbf{c}_{\mathbf{io}\,\Gamma} y)$.

true

$=$ { $xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ }

$xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$

$\Rightarrow$ { **ptr** Γ is prefix-closed, see property 2.45(ii) }

$x \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$

$\Rightarrow$ { induction hypothesis, using $lx < lt$ and calculus }

$(\mathbf{E}\, z : z \in \mathbf{t}(\mathbf{dse}\,\Gamma) : x\mathbf{c}_{\mathbf{io}\,\Gamma} z) \Rightarrow x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

$\Rightarrow$ { property 4.7, using $(\mathbf{E}\, y : y \in \mathbf{t}(\mathbf{dse}\,\Gamma) : xa\mathbf{c}_{\mathbf{io}\,\Gamma} y)$ }

$x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\Gamma$

$$
\begin{array}{ll}
 & \text{true} \\
= & \quad \{\ x \in \mathbf{t}(\mathbf{dse}\Gamma)\ \} \\
 & x \in \mathbf{t}(\mathbf{dse}\Gamma) \\
= & \quad \{\ \text{property 4.6(i), using } a \in \mathbf{o}\Gamma\ \} \\
 & x \in \mathbf{t}(\mathbf{dse}\Gamma) \wedge xa\mathbf{c}_{\mathbf{io}\Gamma}x \\
= & \quad \{\ \text{definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{o}\Gamma \text{ and } xa \in \mathbf{t}(\mathbf{ptr}\Gamma)\ \} \\
 & xa \in \mathbf{t}(\mathbf{dse}\Gamma)
\end{array}
$$

case 1: $a \in \mathbf{i}\Gamma$

We know that $(\mathbf{E}y : y \in \mathbf{t}(\mathbf{dse}\Gamma) : xa\mathbf{c}_{\mathbf{io}\Gamma}y)$. Given such a trace y. Hence $y \in \mathbf{t}(\mathbf{dse}\Gamma)$ and $xa\mathbf{c}_{\mathbf{io}\Gamma}y$. Using $a \in \mathbf{i}\Gamma$ we find that $(\mathbf{E}b, w, z : b \in \mathbf{i}\Gamma \wedge z \in (\mathbf{o}\Gamma)^* : wbz = y)$. Given such a symbol b and such traces w and z. Hence $b \in \mathbf{i}\Gamma$. From property 4.40(ii) we derive that $wb \in \mathbf{t}(\mathbf{dse}\Gamma)$. From property 4.2, "composability", we derive that $xa\mathbf{c}_{\mathbf{io}\Gamma}wb$.

$$
\begin{array}{ll}
 & \text{true} \\
= & \quad \{\ \text{property 4.41, using } wb \in \mathbf{t}(\mathbf{dse}\Gamma)\ \} \\
 & (\mathbf{A}s, c : s \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge c \in \mathbf{i}\Gamma \wedge sc\mathbf{c}_{\mathbf{io}\Gamma}wb : sc \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
\Rightarrow & \quad \{\ \text{predicate calculus}\ \} \\
 & (\mathbf{A}s, c : s \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge c \in \mathbf{i}\Gamma \wedge sc\mathbf{c}_{\mathbf{io}\Gamma}xa \wedge sc\mathbf{c}_{\mathbf{io}\Gamma}wb : sc \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
= & \quad \{\ \text{property 4.10, "transitivity of composability", using } xa\mathbf{c}_{\mathbf{io}\Gamma}wb\ \} \\
 & (\mathbf{A}s, c : s \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge c \in \mathbf{i}\Gamma \wedge sc\mathbf{c}_{\mathbf{io}\Gamma}xa : sc \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
= & \quad \{\ \text{property 4.11(iii)}\ \} \\
 & (\mathbf{A}s, c : s \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge c \in \mathbf{i}\Gamma \wedge sc\mathbf{c}_{\mathbf{io}\Gamma}xa \wedge \#_c s < \#_c xa : sc \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
= & \quad \{\ \text{definition 4.36, "}\mathbf{dse}\text{", using } x \in \mathbf{t}(\mathbf{dse}\Gamma) \text{ and } a \in \mathbf{i}\Gamma\ \} \\
 & xa \in \mathbf{t}(\mathbf{dse}\Gamma)
\end{array}
$$

Hence, $xa \in \mathbf{t}(\mathbf{dse}\Gamma)$; since $xa = t$, we conclude $t \in \mathbf{t}(\mathbf{dse}\Gamma)$.

end of lemma

lemma 4.51

For component Γ,

$$(\mathbf{dse}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4$$

proof

Given component Γ. Let traces t, y, and z be such that $y \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, $z \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, $y\mathbf{c}_{\mathbf{io}\Gamma}t$, and $t\mathbf{c}_{\mathbf{io}\Gamma}z$. We prove by induction on t, that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

induction hypothesis

$$(\mathbf{A}\,u : u\,\mathbf{prefix}\,t \wedge lu < lt : u \in \mathbf{t}(\mathbf{dse}\,\Gamma))$$

base: $lt = 0$

Hence $t = \varepsilon$. From definition 4.36, "**dse**", we derive that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

step: $lt > 0$

Let $t = xa$ for trace x and symbol a; hence, $x\,\mathbf{prefix}\,t$, $lx < lt$, $y\mathbf{c}_{\mathbf{io}\Gamma}xa$, and $xa\mathbf{c}_{\mathbf{io}\Gamma}z$. From the induction hypothesis we conclude that $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\Gamma$

$$\begin{array}{ll} & \text{true} \\ = & \{\ y \in \mathbf{t}(\mathbf{dse}\,\Gamma)\ \} \\ & y \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\ \Rightarrow & \{\ \text{lemma 4.43}\ \} \\ & (\mathbf{E}\,u : u \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : u\mathbf{c}_{\mathbf{io}\Gamma}y) \\ \Rightarrow & \{\ \text{property 4.10, "transitivity of composability", using } y\mathbf{c}_{\mathbf{io}\Gamma}xa\ \} \\ & (\mathbf{E}\,u : u \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : u\mathbf{c}_{\mathbf{io}\Gamma}xa) \\ = & \{\ \text{property 4.11(i), using } a \in \mathbf{o}\Gamma\ \} \\ & (\mathbf{E}\,u : u \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : u\mathbf{c}_{\mathbf{io}\Gamma}x \wedge \#_a u > \#_a x) \\ = & \{\ \text{definition 4.36, "dse", using } a \in \mathbf{o}\Gamma \text{ and } x \in \mathbf{t}(\mathbf{dse}\,\Gamma)\ \} \\ & xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \end{array}$$

case 1: $a \in \mathbf{i}\Gamma$

Since $a \in \mathbf{i}\Gamma$ and $xa\mathbf{c}_{\mathbf{lo}\Gamma}z$ we derive, using property 4.11(iii), that $\#_a z > 0$. Hence, there exist traces v and w, and symbol b such that $z = vbw$, $b \in \mathbf{i}\Gamma$, and $w \in (\mathbf{o}\Gamma)^*$. Given such v, w, and b. Hence, $vbw \in \mathbf{t}(\mathbf{dse}\Gamma)$ and $xa\mathbf{c}_{\mathbf{lo}\Gamma}vbw$. From property 4.40(ii) we derive that $vb \in \mathbf{t}(\mathbf{dse}\Gamma)$. Using $w \in (\mathbf{o}\Gamma)^*$ and $xa\mathbf{c}_{\mathbf{lo}\Gamma}vbw$, we derive from property 4.11(i) that $xa\mathbf{c}_{\mathbf{lo}\Gamma}vb$.

$$
\begin{array}{ll}
 & \text{true} \\
= & \{\ \text{property 4.41, using } b \in \mathbf{i}\Gamma \text{ and } vb \in \mathbf{t}(\mathbf{dse}\Gamma)\ \} \\
 & (\mathbf{A}\, u, c : u \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge c \in \mathbf{i}\Gamma \wedge uc\mathbf{c}_{\mathbf{lo}\Gamma}vb : uc \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
\Rightarrow & \{\ \text{calculus}\ \} \\
 & (\mathbf{A}\, u, c : u \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge c \in \mathbf{i}\Gamma \wedge uc\mathbf{c}_{\mathbf{lo}\Gamma}xa \wedge uc\mathbf{c}_{\mathbf{lo}\Gamma}vb : uc \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
= & \{\ (\mathbf{A}\, s : s \in (\mathbf{a}\Gamma)^* : (sc_{\mathbf{lo}\Gamma}xa \wedge sc_{\mathbf{lo}\Gamma}vb) = sc_{\mathbf{lo}\Gamma}xa), \text{ see} \\
 & \quad \text{property 4.10, "transitivity of composability", using} \\
 & \quad xa\mathbf{c}_{\mathbf{lo}\Gamma}vb \\
 & \} \\
 & (\mathbf{A}\, u, c : u \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge c \in \mathbf{i}\Gamma \wedge uc\mathbf{c}_{\mathbf{lo}\Gamma}xa : uc \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
= & \{\ \text{property 4.11(iii)}\ \} \\
 & (\mathbf{A}\, u, c : u \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge c \in \mathbf{i}\Gamma \wedge u\mathbf{c}_{\mathbf{lo}\Gamma}xa \wedge \#_c u < \#_c xa : uc \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
= & \{\ \text{definition 4.36, "}\mathbf{dse}\text{", using } x \in \mathbf{t}(\mathbf{dse}\Gamma) \text{ and } a \in \mathbf{i}\Gamma\ \} \\
 & xa \in \mathbf{t}(\mathbf{dse}\Gamma)
\end{array}
$$

Hence, $xa \in \mathbf{t}(\mathbf{dse}\Gamma)$; since $xa = t$, we conclude that $t \in \mathbf{t}(\mathbf{dse}\Gamma)$.

We have proven that

$$(\mathbf{A}\, y, t, z : y \in \mathbf{t}(\mathbf{dse}\Gamma) \wedge y\mathbf{c}_{\mathbf{lo}\Gamma}t \wedge t\mathbf{c}_{\mathbf{lo}\Gamma}z \wedge z \in \mathbf{t}(\mathbf{dse}\Gamma) : t \in \mathbf{t}(\mathbf{dse}\Gamma)).$$

Now we conclude from definition 4.16, "$\mathbf{D}_4$", and $\mathbf{ab}\Gamma = \mathbf{o}\Gamma \oplus \mathbf{i}\Gamma$ that $(\mathbf{dse}\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4$.

end of lemma

lemma 4.52

For component Γ,

$$(\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4 = (\mathbf{dse}\,\Gamma = \mathbf{ptr}\,\Gamma)$$

proof

Given component Γ. We first prove that $(\mathbf{dse}\,\Gamma = \mathbf{ptr}\,\Gamma) \Rightarrow (\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4$.

Let component Γ be such that $\mathbf{dse}\,\Gamma = \mathbf{ptr}\,\Gamma$. From lemma 4.51 we conclude that $(\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4$.

We now prove that $(\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4 \Rightarrow (\mathbf{dse}\,\Gamma = \mathbf{ptr}\,\Gamma)$. Let component Γ be such that $(\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4$. Now, $\mathbf{a}(\mathbf{dse}\,\Gamma) = \mathbf{a}(\mathbf{ptr}\,\Gamma)$. For trace t we prove that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma) = t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ by induction on the length of t.

induction hypothesis

$$(\mathbf{A}\,u : lu < lt : u \in \mathbf{t}(\mathbf{dse}\,\Gamma) = u \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$$

base: $lt = 0$

Hence, $t = \varepsilon$. Since $\varepsilon \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ and $\varepsilon \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, we conclude that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma) = t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$.

step: $lt > 0$

Let $t = xa$ for trace x and symbol a; hence, $lx < lt$.

We first prove that $xa \in (\mathbf{dse}\,\Gamma) \Rightarrow xa \in (\mathbf{ptr}\,\Gamma)$.

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\Gamma$

Using $a \in \mathbf{o}\Gamma$ we derive from property 4.6(i) that $xa\,\mathbf{c}_{\mathbf{io}\Gamma}\,x$.

$$\begin{array}{ll} & xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\ = & \quad \{ \text{ definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{o}\Gamma \ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge s\,\mathbf{c}_{\mathbf{io}\Gamma}\,x : \#_a s > \#_a x) \\ = & \quad \{ \text{ induction hypothesis, using } lx < lt \ \} \\ & x \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge s\,\mathbf{c}_{\mathbf{io}\Gamma}\,x : \#_a s > \#_a x) \\ = & \quad \{ \text{ property 4.11(i), using } a \in \mathbf{o}\Gamma \ \} \\ & x \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\,\mathbf{c}_{\mathbf{io}\Gamma}\,xa) \\ = & \quad \{ \text{ definition 4.16, "}\mathbf{D}_4\text{", using } (\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4 \text{ and } xa\,\mathbf{c}_{\mathbf{io}\Gamma}\,x \ \} \\ & x \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \\ = & \quad \{ \text{ property 2.45(ii) } \} \\ & xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \end{array}$$

case 1: $a \in \mathbf{i}\Gamma$

Using $a \in \mathbf{i}\Gamma$ we derive from property 4.6(ii) that $x\mathbf{c}_{\mathbf{io}\Gamma}xa$.

$$
\begin{array}{ll}
 & xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\
= & \quad \{ \text{ definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{i}\Gamma \ \} \\
 & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\
 & \wedge (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge s\mathbf{c}_{\mathbf{io}\Gamma}xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)) \\
= & \quad \{ \text{ induction hypothesis, using } lx < lt \ \} \\
 & x \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \\
 & \wedge (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge s\mathbf{c}_{\mathbf{io}\Gamma}xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)) \\
\Rightarrow & \quad \{ \ a \in \mathbf{i}\Gamma,\ x\mathbf{c}_{\mathbf{io}\Gamma}xa, \text{ and calculus } \} \\
 & xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)
\end{array}
$$

Hence, $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \Rightarrow xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$.

We now prove that $xa \in (\mathbf{ptr}\,\Gamma) \Rightarrow xa \in (\mathbf{dse}\,\Gamma)$.

$$
\begin{array}{ll}
 & xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \\
= & \quad \{ \text{ property 2.45(ii) } \} \\
 & x \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \\
= & \quad \{ \text{ induction hypothesis, using } lx < lt \ \} \\
 & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \\
= & \quad \{ \ (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma : s\mathbf{c}_{\mathbf{io}\Gamma}sb), \text{ see property 4.6(ii) } \} \\
 & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma : s\mathbf{c}_{\mathbf{io}\Gamma}sb) \\
\Rightarrow & \quad \{ \text{ definition 4.16, "}\mathbf{D}_4\text{", using } (\mathbf{ptr}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4 \ \} \\
 & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma : sb\mathbf{c}_{\mathbf{io}\Gamma}xa \Rightarrow sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)) \\
= & \quad \{ \text{ predicate calculus and property 4.11(iii) } \} \\
 & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\
 & \wedge (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge s\mathbf{c}_{\mathbf{io}\Gamma}xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)) \\
= & \quad \{ \text{ definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{i}\Gamma \ \} \\
 & xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)
\end{array}
$$

Hence, we have proven that $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) = xa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$; since $xa = t$, we conclude that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma) = t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$.

end of lemma

theorem 4.56

For i/o-connectable components Γ and Δ,

$$\Gamma \; \textit{NCIHDS} \; \Delta = (\mathbf{DSE}\,\Gamma) \; \textit{NCIH} \; (\mathbf{DSE}\,\Delta)$$

proof

Given i/o-connectable components Γ and Δ.

$\Gamma \; \textit{NCIHDS} \; \Delta$
$=$ { property 4.31, "symmetry of *NCIHDS*" }
$\Delta \; \textit{NCIHDS} \; \Gamma$
$=$ { property 4.54(iii) }
$(\mathbf{DSE}\,\Delta) \; \textit{NCIHDS} \; \Gamma$
$=$ { property 4.31, "symmetry of *NCIHDS*" }
$\Gamma \; \textit{NCIHDS} \; (\mathbf{DSE}\,\Delta)$
$=$ { property 4.55(iii) }
$(\mathbf{DSE}\,\Gamma) \; \textit{NCIH} \; (\mathbf{DSE}\,\Delta)$

end of theorem

lemma 4.69

For component Γ,

$$(\mathbf{CBDS}\,\Gamma)\ \textit{NCIHDS}\ \overline{\mathbf{DSE}\,\Gamma}$$

proof

Given component Γ. From definition 4.34, "delay-safe enclosure", we infer that $\Gamma\ \textit{NCIHADS}\ \overline{\mathbf{DSE}\,\Gamma}$. Using that $\mathbf{ptr}(\mathbf{CBDS}\,\Gamma)\subseteq \mathbf{ptr}\,\Gamma$, see definition 4.68, "maximal communication behavior for delay-safe communication", we conclude that $(\mathbf{CBDS}\,\Gamma)\ \textit{NCIHADS}\ \overline{\mathbf{DSE}\,\Gamma}$.

Let traces t and u and symbol a be such that $t\in \mathbf{t}(\mathbf{ptr}(\mathbf{DSE}\,\Gamma))$, $u\in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma))$, $a\in \mathbf{i}\Gamma$, $t\mathbf{c}_{\mathbf{lo}\overline{\Gamma}}u$, and $\#_a t > \#_a u$. From definition 4.68 we infer that $u\in \mathbf{t}(\mathbf{ptr}\,\Gamma)$. Using definition 4.34, "delay-safe enclosure", we conclude that $ua\in \mathbf{t}(\mathbf{ptr}\,\Gamma)$. From property 4.9, we get that $ua\mathbf{c}_{\mathbf{lo}\Gamma}t$, where $t\in \mathbf{t}(\mathbf{ptr}(\mathbf{DSE}\,\Gamma))$. Using the maximality (w.r.t. trace structure inclusion) of **CBDS**, we conclude that $ua\in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma))$. Now, from definition 4.29, "*NCIHADS*", we infer that $\overline{\mathbf{DSE}\,\Gamma}\ \textit{NCIHADS}\ (\mathbf{CBDS}\,\Gamma)$.

From definition 4.30, "computation interference hazard for indirect connection", we conclude that $(\mathbf{CBDS}\,\Gamma)\ \textit{NCIHDS}\ \overline{\mathbf{DSE}\,\Gamma}$.

end of lemma

theorem 4.74 *maximal communication behavior for delay-safe communication*

For component Γ,

$$\mathbf{ptr}(\mathbf{CBDS}\,\Gamma) = \mathbf{cbds}\,\Gamma$$

proof

Given component Γ. From property 4.72 we derive that $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Gamma$. From definition 4.70, "**cbds**", we derive that $(\mathbf{A}\,a, s : a \in \mathbf{i}\Gamma \wedge sa \in \mathbf{t}(\mathbf{cbds}\,\Gamma) : (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{dse}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\Gamma}t))$.

We now have to prove that $\mathbf{cbds}\,\Gamma$ is maximal. Let trace s and symbol a be such that $s \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$, $sa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, and $a \in \mathbf{i}\Gamma \Rightarrow (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{dse}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\Gamma}t)$. We prove that $sa \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$.

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\Gamma$

$$\begin{array}{ll} & \text{true} \\ = & \{\ \text{property 4.72, using } s \in \mathbf{t}(\mathbf{cbds}\,\Gamma)\ \} \\ & s \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\ = & \{\ \text{property 4.6(i), using } a \in \mathbf{o}\Gamma\ \} \\ & s \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge sa\mathbf{c}_{\mathbf{io}\Gamma}s \\ \Rightarrow & \{\ \text{definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{o}\Gamma \text{ and } sa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)\ \} \\ & sa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\ = & \{\ \text{property 4.72, using } sa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)\ \} \\ & sa \in \mathbf{t}(\mathbf{cbds}\,\Gamma) \end{array}$$

case 1: $a \in \mathbf{i}\Gamma$

$$\begin{array}{ll} & \text{true} \\ = & \{\ (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{dse}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\Gamma}t), \text{ since } a \in \mathbf{i}\Gamma\ \} \\ & (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{dse}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\Gamma}t) \\ = & \{\ \text{definition 4.70, "}\mathbf{cbds}\text{", using } sa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)\ \} \\ & sa \in \mathbf{t}(\mathbf{cbds}\,\Gamma) \end{array}$$

end of theorem

The proof of theorem 4.77 is spread over two pages. It starts at page 240.

theorem 4.77

For components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\overline{\Delta}$,

$$\begin{array}{l}\Gamma\ \mathit{NCIHDS}\ \Delta \\ \Rightarrow\ (\mathbf{A}\,t,u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\,\Gamma}\,u \wedge u \in \mathbf{t}(\mathbf{ptr}\,\Delta) \\ \qquad\qquad : t \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma)) \wedge u \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Delta)) \\ \quad)\end{array}$$

proof

Given components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\overline{\Delta}$, Let traces t and u be such that, $t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, $t\mathbf{c}_{\mathbf{io}\,\Gamma}\,u$, and $u \in \mathbf{t}(\mathbf{ptr}\,\Delta)$.

We first prove that $t \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma))$. In order to be able to prove this, we prove that $u \in \mathbf{t}(\mathbf{dse}\,\Gamma)$. The latter we prove by induction on the length of u.

induction hypothesis

$$(\mathbf{A}\,v : v \in \mathbf{t}(\mathbf{ptr}\,\Delta) \wedge lv < lu \wedge (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,v) : v \in \mathbf{t}(\mathbf{dse}\,\Gamma))$$

base: $lu = 0$

Hence $u = \varepsilon$. From property 4.40(i), we conclude that $u \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

step: $lu > 0$

Let $u = xa$ for trace x and symbol a; hence, $x\mathbf{prefix}u$, $lx < lu$, and $t\mathbf{c}_{\mathbf{io}\,\Gamma}\,xa$. We prove that $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

$$\begin{array}{ll} & \mathbf{true} \\ = & \{\ x\mathbf{prefix}u \text{ and property 4.7, using } t\mathbf{c}_{\mathbf{io}\,\Gamma}\,u\ \} \\ & (\mathbf{E}\,s : s\mathbf{prefix}t : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,x) \wedge x\mathbf{prefix}u \\ \Rightarrow & \{\ \text{property 2.45(ii) twice, using } t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \text{ and } u \in \mathbf{t}(\mathbf{ptr}\,\Delta)\ \} \\ & (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,x) \wedge x \in \mathbf{t}(\mathbf{ptr}\,\Delta) \\ \Rightarrow & \{\ \text{induction hypothesis, using } lx < lu\ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \end{array}$$

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\Gamma$

$$
\begin{array}{ll}
 & \text{true} \\
= & \quad \{ \text{ property 4.11(i), using } a \in \mathbf{o}\Gamma \text{ and } t\mathbf{c}_{\mathbf{io}\Gamma} xa \ \} \\
 & t\mathbf{c}_{\mathbf{io}\Gamma} x \wedge \#_a t > \#_a x \\
\Rightarrow & \quad \{ \text{ definition 4.36, "}\mathbf{dse}\text{", using } x \in \mathbf{t}(\mathbf{dse}\Gamma) \text{ and } a \in \mathbf{o}\Gamma \ \} \\
 & xa \in \mathbf{t}(\mathbf{dse}\Gamma)
\end{array}
$$

case 1: $a \in \mathbf{i}\Gamma$

$$
\begin{array}{ll}
 & \text{true} \\
= & \quad \{ \text{ definition 4.30, using } \Gamma \ \textit{NCIHDS}\ \Delta, \text{ and definition 4.29,} \\
 & \quad \text{using } xa \in \mathbf{t}(\mathbf{ptr}\,\Delta) \\
 & \quad \} \\
 & (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge s\mathbf{c}_{\mathbf{io}\Gamma} xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)) \\
= & \quad \{ \text{ definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{i}\Gamma \text{ and } x \in \mathbf{t}(\mathbf{dse}\Gamma) \ \} \\
 & xa \in \mathbf{t}(\mathbf{dse}\Gamma)
\end{array}
$$

Hence, we have proven that $xa \in \mathbf{t}(\mathbf{dse}\Gamma)$; since $xa = u$ we conclude that $u \in \mathbf{t}(\mathbf{dse}\Gamma)$.

We have proved that $u \in \mathbf{t}(\mathbf{dse}\Gamma)$. Now, using $t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $t\mathbf{c}_{\mathbf{io}\Gamma} u$, we infer from definition 4.70, "**cbds**", that $t \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$. From theorem 4.74, we now conclude that $t \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma))$.

We now have proven that for components Γ and Δ such that $\mathbf{io}\Gamma = \mathbf{io}\,\overline{\Delta}$, $\Gamma \ \textit{NCIHDS}\ \Delta \Rightarrow (\mathbf{A}\, t, u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\Gamma} u \wedge u \in \mathbf{t}(\mathbf{ptr}\,\Delta) : t \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Gamma)))$.

Using that $\mathbf{io}\Delta = \mathbf{io}\,\overline{\Gamma}$, and $\Gamma \ \textit{NCIHDS}\ \Delta = \Delta \ \textit{NCIHDS}\ \Gamma$, we conclude that $\Gamma \ \textit{NCIHDS}\ \Delta \Rightarrow (\mathbf{A}\, t, u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\Gamma} u \wedge u \in \mathbf{t}(\mathbf{ptr}\,\Delta) : u \in \mathbf{t}(\mathbf{ptr}(\mathbf{CBDS}\,\Delta)))$.

end of theorem

theorem 4.80

For components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$, $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$, and $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$,

$$\mathbf{dse}\,\Gamma = \mathbf{dse}\,\Delta$$

proof

Given components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$, $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$, and $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$.

We first prove that $\mathbf{dse}\,\Gamma \subseteq \mathbf{dse}\,\Delta$. Let trace t be such that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

induction hypothesis

$$(\mathbf{A}\,u : u \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge lu < lt : u \in \mathbf{t}(\mathbf{dse}\,\Delta))$$

base: $lt = 0$

Hence $t = \varepsilon$. From property 4.40(i) we conclude that $t \in \mathbf{t}(\mathbf{dse}\,\Delta)$.

step: $lt > 0$

Let $t = xa$ for trace x and symbol a; hence, $lx < lt$ and $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$. From property 4.40(ii) we conclude that $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$. From the induction hypothesis we infer that $x \in \mathbf{t}(\mathbf{dse}\,\Delta)$.

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\,\Gamma$ and $\mathbf{i}\,\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\,\Gamma$

From $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$ we infer that $a \in \mathbf{o}\,\Delta$.

true
$=$ { definition 4.36, "dse", using $a \in \mathbf{o}\,\Gamma$ and $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ }
$(\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,xa)$
$=$ { property 4.5, using $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ }
$(\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{cbds}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,xa)$
$\Rightarrow$ { $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$ }
$(\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Delta) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,xa)$
$=$ { definition 4.36, "dse", using $x \in \mathbf{t}(\mathbf{dse}\,\Delta)$ and $a \in \mathbf{o}\,\Delta$ }
$xa \in \mathbf{t}(\mathbf{dse}\,\Delta)$

case 1: $a \in \mathbf{i}\Gamma$

From $\mathbf{io}\Gamma = \mathbf{io}\Delta$ we infer that $a \in \mathbf{i}\Delta$.

$$
\begin{array}{ll}
 & \text{true} \\
= & \quad \{ \text{ definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{i}\Gamma \text{ and } xa \in \mathbf{t}(\mathbf{dse}\Gamma) \ \} \\
 & (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge s\mathbf{c}_{\mathbf{io}\Gamma} xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
\Rightarrow & \quad \{ \ \mathbf{ptr}\Delta \subseteq \mathbf{ptr}\Gamma \text{ and } \mathbf{io}\Gamma = \mathbf{io}\Delta \ \} \\
 & (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\Delta) \wedge b \in \mathbf{i}\Delta \wedge s\mathbf{c}_{\mathbf{io}\Delta} xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\Gamma)) \\
= & \quad \{ \text{ definition 4.70, "}\mathbf{cbds}\text{", using } xa \in \mathbf{t}(\mathbf{dse}\Gamma) \ \} \\
 & (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\Delta) \wedge b \in \mathbf{i}\Delta \wedge s\mathbf{c}_{\mathbf{io}\Delta} xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{cbds}\Gamma)) \\
\Rightarrow & \quad \{ \ \mathbf{cbds}\Gamma \subseteq \mathbf{ptr}\Delta \ \} \\
 & (\mathbf{A}\, s, b : s \in \mathbf{t}(\mathbf{ptr}\Delta) \wedge b \in \mathbf{i}\Delta \wedge s\mathbf{c}_{\mathbf{io}\Delta} xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\Delta)) \\
= & \quad \{ \text{ definition 4.36, "}\mathbf{dse}\text{", using } x \in \mathbf{t}(\mathbf{dse}\Delta) \text{ and } a \in \mathbf{i}\Delta \ \} \\
 & xa \in \mathbf{t}(\mathbf{dse}\Delta)
\end{array}
$$

Hence, we have proven that $xa \in \mathbf{t}(\mathbf{dse}\Delta)$; since $t = xa$, we conclude that $t \in \mathbf{t}(\mathbf{dse}\Delta)$.

We conclude that $\mathbf{dse}\Gamma \subseteq \mathbf{dse}\Delta$.

We now prove that $\mathbf{dse}\Delta \subseteq \mathbf{dse}\Gamma$. Let trace t be such that $t \in \mathbf{t}(\mathbf{dse}\Delta)$. We prove that $t \in \mathbf{t}(\mathbf{dse}\Gamma)$ by mathematical induction on $l(t \restriction \mathbf{i}\Delta)$.

induction hypothesis

$$(\mathbf{A}\, u : u \in \mathbf{t}(\mathbf{dse}\Delta) \wedge l(u \restriction \mathbf{i}\Delta) < l(t \restriction \mathbf{i}\Delta) : u \in \mathbf{t}(\mathbf{dse}\Gamma))$$

base: $l(t \restriction \mathbf{i}\Delta) = 0$

Since $\mathbf{a}\Delta$ is bipartitioned into $\mathbf{o}\Delta$ and $\mathbf{i}\Delta$, we conclude that $t \in (\mathbf{o}\Delta)^*$. From property 4.40(i) we conclude that $\varepsilon \in \mathbf{t}(\mathbf{dse}\Gamma)$. Now we infer from property 4.79 that $t \in \mathbf{t}(\mathbf{dse}\Gamma)$.

step: $l(t \restriction \mathbf{i}\Delta) > 0$

Let $t = xay$ for traces x and y and symbol a such that $a \in \mathbf{i}\Delta$, and $y \in (\mathbf{o}\Delta)^*$; hence, $l(x \restriction \mathbf{i}\Delta) < l(t \restriction \mathbf{i}\Delta)$ and $xay \in \mathbf{t}(\mathbf{dse}\Delta)$. From property 4.40(ii) we conclude that $xa \in \mathbf{t}(\mathbf{dse}\Delta)$ and $x \in \mathbf{t}(\mathbf{dse}\Delta)$. From the induction hypothesis we infer that $x \in \mathbf{t}(\mathbf{dse}\Gamma)$. From $\mathbf{io}\Gamma = \mathbf{io}\Delta$ we infer that $a \in \mathbf{i}\Gamma$ and $y \in (\mathbf{o}\Gamma)^*$.

Let trace w and symbol b be such that $w \in \mathbf{t}(\mathbf{ptr}\Gamma)$, $b \in \mathbf{i}\Gamma$, $w\mathbf{c}_{\mathbf{io}\Gamma} xa$, and $\#_b w < \#_b xa$. From property 4.11(iii) we infer that $wb\mathbf{c}_{\mathbf{io}\Gamma} xa$. Since $\mathbf{io}\Gamma = \mathbf{io}\Delta$, we infer that $b \in \mathbf{i}\Delta$ and $wb\mathbf{c}_{\mathbf{io}\Delta} xa$.

Suppose that $w \notin \mathbf{t}(\mathbf{ptr}\,\Delta)$.

Using definition 4.70, "**cbds**", we infer that $\mathbf{ptr}\,\Gamma \cap \mathbf{dse}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$. Since $w \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, we conclude that $w \notin \mathbf{t}(\mathbf{dse}\,\Gamma)$. Now, using properties 4.40(i) and 4.40(ii) we derive, that there (uniquely) exist traces u and v and symbol c such that $w = ucv$, $u \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, and

$$uc \notin \mathbf{t}(\mathbf{dse}\,\Gamma). \qquad (*0)$$

Given such u, v, and c. Hence, $ucvb\mathbf{c}_{\mathbf{io}\,\Delta}xa$ and, using property 4.40(ii), $uc \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $u \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$.

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $c \in \mathbf{o}\Gamma$

$$\begin{array}{ll} & \mathrm{true} \\ = & \{\ uc \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \text{ and property 4.6(i) using } c \in \mathbf{o}\Gamma\ \} \\ & uc \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge uc\mathbf{c}_{\mathbf{io}\,\Gamma}u \\ \Rightarrow & \{\ \text{definition 4.36, "}\mathbf{dse}\text{", using } u \in \mathbf{t}(\mathbf{dse}\,\Gamma) \text{ and } c \in \mathbf{o}\Gamma\ \} \\ & uc \in \mathbf{t}(\mathbf{dse}\,\Gamma) \end{array}$$

case 1: $c \in \mathbf{i}\Gamma$

First we derive:

$$\begin{array}{ll} & l(uc \restriction \mathbf{i}\Delta) \\ < & \{\ b \in \mathbf{i}\Delta\ \} \\ & l(ucvb \restriction \mathbf{i}\Delta) \\ \leq & \{\ \text{property 4.2, "composability", using } ucvb\mathbf{c}_{\mathbf{io}\,\Delta}xa\ \} \\ & l(xa \restriction \mathbf{i}\Delta) \\ = & \{\ t = xay \text{ and } y \in (\mathbf{o}\Delta)^*\ \} \\ & l(t \restriction \mathbf{i}\Delta) \end{array}$$

Hence, $l(uc \restriction \mathbf{i}\Delta) < l(t \restriction \mathbf{i}\Delta)$.

From (several times) property 4.7, using that $ucvb\mathbf{c}_{\mathbf{io}\,\Delta}xa$, we derive that $(\mathbf{E}\,z : z\,\mathbf{prefix}\,xa : uc\mathbf{c}_{\mathbf{io}\,\Delta}z)$. Using $xa \in \mathbf{t}(\mathbf{dse}\,\Delta)$ and property 4.40(ii), we conclude that $(\mathbf{E}\,z : z \in \mathbf{t}(\mathbf{dse}\,\Delta) : uc\mathbf{c}_{\mathbf{io}\,\Delta}z)$.

true
= { property 4.72, using $u \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $u \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ }
$u \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$
⇒ { $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$ }
$u \in \mathbf{t}(\mathbf{ptr}\,\Delta)$
= { $(\mathbf{E}\,z : z \in \mathbf{t}(\mathbf{dse}\,\Delta) : uc\mathbf{c}_{\mathbf{io}\,\Delta}z)$ }
$u \in \mathbf{t}(\mathbf{ptr}\,\Delta) \wedge (\mathbf{E}\,z : z \in \mathbf{t}(\mathbf{dse}\,\Delta) : uc\mathbf{c}_{\mathbf{io}\,\Delta}z)$
⇒ { property 4.41, using $c \in \mathbf{i}\Gamma$ }
$uc \in \mathbf{t}(\mathbf{ptr}\,\Delta) \wedge (\mathbf{E}\,z : z \in \mathbf{t}(\mathbf{dse}\,\Delta) : uc\mathbf{c}_{\mathbf{io}\,\Delta}z)$
⇒ { lemma 4.49 }
$uc \in \mathbf{t}(\mathbf{dse}\,\Delta)$
⇒ { induction hypothesis, using $l(uc \restriction \mathbf{i}\Delta) < l(t \restriction \mathbf{i}\Delta)$ }
$uc \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

We have proven that $uc \in \mathbf{t}(\mathbf{dse}\,\Gamma)$. This is in contradiction with (***0**).

Hence, $w \in \mathbf{t}(\mathbf{ptr}\,\Delta)$.

true
= { definition 4.36, "**dse**", using $xa \in \mathbf{t}(\mathbf{dse}\,\Delta)$ and $a \in \mathbf{i}\Delta$ }
$(\mathbf{A}\,r, d : r \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge d \in \mathbf{i}\Gamma \wedge r\mathbf{c}_{\mathbf{io}\,\Gamma}xa \wedge \#_d r < \#_d xa : rd \in \mathbf{t}(\mathbf{ptr}\,\Gamma))$
⇒ { $w \in \mathbf{t}(\mathbf{ptr}\,\Delta)$, $b \in \mathbf{i}\Delta$, and $wb\mathbf{c}_{\mathbf{io}\,\Delta}xa$ }
$wb \in \mathbf{t}(\mathbf{ptr}\,\Delta)$
⇒ { $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$ }
$wb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$

Now we have proven that

$$(\mathbf{A}\,w, b : w \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge w\mathbf{c}_{\mathbf{io}\,\Gamma}xa \wedge \#_b w < \#_b xa : wb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)).$$

From definition 4.36, "**dse**", using $x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ and $a \in \mathbf{i}\Gamma$, we derive that $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$. From property 4.79, using $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$, $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$, $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, $y \in (\mathbf{o}\,\Gamma)^*$, and $xay \in \mathbf{t}(\mathbf{dse}\,\Delta)$, we derive that $xay \in \mathbf{t}(\mathbf{dse}\,\Gamma)$. Since $t = xay$, we conclude that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

Hence, $\mathbf{dse}\,\Delta \subseteq \mathbf{dse}\,\Gamma$.

end of theorem

lemma 4.82 **CBDS** *is idempotent*

For component Γ,

$$\mathbf{CBDS}\,(\mathbf{CBDS}\,\Gamma) = \mathbf{CBDS}\,\Gamma$$

proof

Given component Γ. We notice that $\mathbf{a}(\mathbf{CBDS}\,(\mathbf{CBDS}\,\Gamma)) = \mathbf{a}\Gamma$ and $\mathbf{a}(\mathbf{CBDS}\,\Gamma) = \mathbf{a}\Gamma$.

$\mathbf{ptr}\,(\mathbf{CBDS}\,(\mathbf{CBDS}\,\Gamma))$
$=$ { theorem 4.74 }
$\mathbf{cbds}\,(\mathbf{CBDS}\,\Gamma)$
$=$ { property 4.72 }
$\mathbf{ptr}\,(\mathbf{CBDS}\,\Gamma) \cap \mathbf{dse}\,(\mathbf{CBDS}\,\Gamma)$
$=$ { theorem 4.80 }
$\mathbf{ptr}\,(\mathbf{CBDS}\,\Gamma) \cap \mathbf{dse}\,\Gamma$
$=$ { theorem 4.74 }
$\mathbf{cbds}\,\Gamma \cap \mathbf{dse}\,\Gamma$
$=$ { property 4.72 }
$\mathbf{ptr}\,\Gamma \cap \mathbf{dse}\,\Gamma \cap \mathbf{dse}\,\Gamma$
$=$ { calculus }
$\mathbf{ptr}\,\Gamma \cap \mathbf{dse}\,\Gamma$
$=$ { property 4.72 }
$\mathbf{cbds}\,\Gamma$
$=$ { theorem 4.74 }
$\mathbf{ptr}\,(\mathbf{CBDS}\,\Gamma)$

end of lemma

lemma 4.84

For component Γ, no input comminst enables an input comminst in $\mathbf{CBDS}\,\Gamma$.

proof

Given component Γ. Let trace t and symbols a and b be such that $a \in \mathbf{i}\Gamma$, $b \in \mathbf{i}\Gamma$, $ta \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$, and $tab \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$.

true

= { property 4.2, "composability", using $a \in \mathbf{i}\Gamma$ }

$tb\,\mathbf{c}_{\mathbf{io}\,\Gamma}\,tab$

= { $t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $tab \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, see property 4.72, using property 2.45(ii) }

$t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\,\mathbf{c}_{\mathbf{io}\,\Gamma}\,tab \wedge tab \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

$\Rightarrow$ { property 4.41, using $b \in \mathbf{i}\Gamma$ }

$tb \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge tb\,\mathbf{c}_{\mathbf{io}\,\Gamma}\,tab \wedge tab \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

$\Rightarrow$ { definition 4.70, "**cbds**" }

$tb \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$

Hence, no input comminst may enable an input comminst in $\mathbf{CBDS}\,\Gamma$.

end of lemma

lemma 4.85

For component Γ, no output comminst disables an input comminst in $\mathbf{CBDS}\,\Gamma$.

proof

Given component Γ. Let trace t and symbols a and b be such that $a \in \mathbf{i}\Gamma$, $b \in \mathbf{o}\Gamma$, $ta \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$, and $tb \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$.

true

= { property 4.2, "composability", using $b \in \mathbf{o}\Gamma$ }

$tba\mathbf{c}_{\mathbf{io}\Gamma}ta$

= { $tb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $ta \in \mathbf{t}(\mathbf{dse}\,\Gamma)$, see property 4.72 }

$tb \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge tba\mathbf{c}_{\mathbf{io}\Gamma}ta \wedge ta \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

$\Rightarrow$ { property 4.41, using $a \in \mathbf{i}\Gamma$ }

$tba \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge tba\mathbf{c}_{\mathbf{io}\Gamma}ta \wedge ta \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

$\Rightarrow$ { definition 4.70, "**cbds**" }

$tba \in \mathbf{t}(\mathbf{cbds}\,\Gamma)$

Hence, no output comminst may disable an input comminst in $\mathbf{CBDS}\,\Gamma$.

end of lemma

A.5 Communicating delay-insensitively

In this section we present the proofs of the lemmas and theorems of chapter 5.

theorem 5.2 $\mathbf{C}_4$

For trace structure T and alphbip D,

$$(T,D)\in\mathbf{C}_4 \;=\; (T,D)\in\mathbf{D}_4 \wedge (\mathbf{A}\,s,a : s\in(\mathbf{a}T)^* \wedge a\in\mathbf{a}T : saa\notin\mathbf{t}T).$$

proof

Given trace structure T and alphbip D.

We first prove that
$(T,D)\in\mathbf{C}_4 \;\Rightarrow\; (T,D)\in\mathbf{D}_4 \wedge (\mathbf{A}\,s,a : s\in(\mathbf{a}T)^* \wedge a\in\mathbf{a}T : saa\notin\mathbf{t}T)$.

Let $(T,D)\in\mathbf{C}_4$. From definition 5.1, "$\mathbf{C}_4$", we conclude that $(T,D)\in\mathbf{D}_4$. Let trace s and symbol a be such that $s\in(\mathbf{a}T)^*$ and $a\in\mathbf{a}T$.

true
= { definition 5.1, "$\mathbf{C}_4$", using $(T,D)\in\mathbf{C}_4$, $s\in(\mathbf{a}T)^*$, and $a\in\mathbf{a}T$ }
$(\mathbf{A}\,t : t\in(\mathbf{a}T)^* \wedge sata\in\mathbf{t}T : I(t\restriction\mathbf{opa}(a,D))>0)$
= { predicate calculus }
$(\mathbf{A}\,t : t\in(\mathbf{a}T)^* : sata\notin\mathbf{t}T \vee (I(t\restriction\mathbf{opa}(a,D))>0))$
⇒ { instantiation, using $\varepsilon\in(\mathbf{a}T)^*$ }
$saa\notin\mathbf{t}T \vee (I(\varepsilon\restriction\mathbf{opa}(a,D))>0)$
⇒ { definition 1.13, "projection of trace" }
$saa\notin\mathbf{t}T$

Hence, $(T,D)\in\mathbf{C}_4 \;\Rightarrow\; ((T,D)\in\mathbf{D}_4 \wedge (\mathbf{A}\,s,a : s\in(\mathbf{a}T)^* \wedge a\in\mathbf{a}T : saa\notin\mathbf{t}T))$.

We now prove that
$((T,D)\in \mathbf{D}_4 \wedge (\mathbf{A}\, s,a : s\in(\mathbf{a}T)^* \wedge a\in \mathbf{a}T : saa\notin \mathbf{t}T)) \Rightarrow (T,D)\in \mathbf{C}_4$.
Let $(T,D)\in \mathbf{D}_4$ and $(\mathbf{A}\, z,c : z\in(\mathbf{a}T)^* \wedge c\in \mathbf{a}T : zcc\notin \mathbf{t}T)$. Let traces s and t, and symbol a be such that $s\in(\mathbf{a}T)^*$, $t\in(\mathbf{a}T)^*$, $a\in \mathbf{a}T$, and $sata\in \mathbf{t}T$. Now, $saa\notin \mathbf{t}T$. Let iobip F be such that $\mathbf{o}F=\mathbf{spa}(a,D)$ and $\mathbf{i}F=\mathbf{opa}(a,D)$; hence, $\mathbf{a}T=\mathbf{o}F\cup\mathbf{i}F$, $a\in\mathbf{o}F$, $D=\mathbf{o}F\oplus\mathbf{i}F$, and $s\in(\mathbf{a}F)^*$.

true
= { property 4.6(i), using $s\in(\mathbf{a}F)^*$ and $a\in\mathbf{a}F$ }
$t\in(\mathbf{o}F)^* \Rightarrow sat\mathbf{c}_F sa$
= { example 4.60, using $s\in(\mathbf{a}F)^*$ and $a\in\mathbf{a}F$ }
$t\in(\mathbf{o}F)^* \Rightarrow sata\mathbf{c}_F saa$
= { $t\in(\mathbf{a}T)^*$, $\mathbf{a}T=\mathbf{o}F\cup\mathbf{i}F$, and $\mathbf{o}F\cap\mathbf{i}F=\varnothing$ }
$(I(t\restriction \mathbf{i}F)=0) \Rightarrow sata\mathbf{c}_F saa$
= { $\mathbf{i}F=\mathbf{opa}(a,D)$ }
$(I(t\restriction \mathbf{opa}(a,D))=0) \Rightarrow sata\mathbf{c}_F saa$
= { property 4.6(i), using $a\in\mathbf{o}F$ }
$(I(t\restriction \mathbf{opa}(a,D))=0) \Rightarrow (sata\mathbf{c}_F saa \wedge saa\mathbf{c}_F s)$
⇒ { definition 4.16, "$\mathbf{D}_4$", using $(T,D)\in\mathbf{D}_4$, $D=\mathbf{o}F\oplus\mathbf{i}F$, and $saa\notin\mathbf{t}T$ }
$(I(t\restriction \mathbf{opa}(a,D))=0) \Rightarrow (sata\notin \mathbf{t}T \vee s\notin \mathbf{t}T)$
= { $\mathbf{t}T$ is prefix-closed, since $(T,D)\in\mathbf{D}_4$, see definition 4.16, "$\mathbf{D}_4$" }
$(I(t\restriction \mathbf{opa}(a,D))=0) \Rightarrow sata\notin \mathbf{t}T$
= { predicate calculus, using $sata\in\mathbf{t}T$ }
$I(t\restriction \mathbf{opa}(a,D))>0$

Hence, $((T,D)\in \mathbf{D}_4 \wedge (\mathbf{A}\, s,a : s\in(\mathbf{a}T)^* \wedge a\in\mathbf{a}T : saa\notin \mathbf{t}T)) \Rightarrow (T,D)\in \mathbf{C}_4$.
end of theorem

lemma 5.16

For di-initializable component Γ,

$$(\mathbf{A}\, t, u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\,\Gamma} u \wedge u \in \mathbf{t}(\mathbf{die}\,\Gamma) : t \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))).$$

proof

Given di-initializable component Γ. Let traces t and u be such that $t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, $t\mathbf{c}_{\mathbf{io}\,\Gamma} u$, and $u \in \mathbf{t}(\mathbf{die}\,\Gamma)$.

Suppose $t \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$

From property 4.40 we derive that there exist a trace v and a symbol b such that $v \in (\mathbf{a}\Gamma)^*$, $b \in \mathbf{a}\Gamma$, $vb\mathbf{prefix}\, t$, $v \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$, and $vb \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$. Given such v and b. Now, using $t\mathbf{c}_{\mathbf{io}\,\Gamma} u$, we derive from property 4.7 that there exists a trace w such that $w\mathbf{prefix}\, u$ and $vb\mathbf{c}_{\mathbf{io}\,\Gamma} w$. Given such w. We first derive that $b \in \mathbf{o}\Gamma$.

true
= { $u \in \mathbf{t}(\mathbf{die}\,\Gamma)$ }
$u \in \mathbf{t}(\mathbf{die}\,\Gamma)$
= { definition 5.14, "**die**" }
$u \in \mathbf{t}(\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma))$
⇒ { property 4.40(ii), using $w\mathbf{prefix}\, u$ }
$w \in \mathbf{t}(\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma))$
⇒ { property 4.41, using $v \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$, $vb\mathbf{c}_{\mathbf{io}\,\Gamma} w$, and $vb \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$ }
$b \notin \mathbf{i}\Gamma$
⇒ { $b \in \mathbf{a}\Gamma$ and $tsa\Gamma = \mathbf{o}\Gamma \cup \mathbf{i}\Gamma$ }
$b \in \mathbf{o}\Gamma$

We have derived that $b \in \mathbf{o}\Gamma$.

true

= { definition 5.11, "**DSENTIH**", using $v \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$ }

$v \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

= { property 2.34(ii), using $t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $vb\,\mathbf{prefix}\,t$ }

$vb \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge v \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

⇒ { definition 4.36, "**dse**", using $b \in \mathbf{o}\,\Gamma$ and $vb\mathbf{c}_{\mathbf{lo}\Gamma}v$, see property 4.6(i) }

$vb \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

= { $vb \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$ }

$vb \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge vb \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$

⇒ { definition 5.11, "**DSENTIH**", and definition 1.34, "**redts**", using $b \in \mathbf{o}\,\Gamma$ }

$v \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$

= { $v \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$ }

false

This is a contradiction.

Hence, $t \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$.

end of lemma

lemma 5.17

For di-initializable component Γ,

$$\mathbf{die}\,\Gamma \subseteq \mathbf{dse}\,\Gamma .$$

proof

Given di-initializable component Γ. Let trace t be such that $t \in \mathbf{t}(\mathbf{die}\,\Gamma)$. We prove that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ by induction on the length of t.

induction hypothesis

$(\mathbf{A}\, u : u \in \mathbf{t}(\mathbf{die}\,\Gamma) \wedge lu < lt : u \in \mathbf{t}(\mathbf{dse}\,\Gamma))$

base: $lt = 0$

Hence $t = \varepsilon$. From property 4.40(i) we infer that $\varepsilon \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

step: $lt > 0$

Let $t = xa$ for trace x and symbol a; hence, $x \in (\mathbf{a}\Gamma)^*$, $a \in \mathbf{a}\Gamma$, $lx < lt$, and $xa \in \mathbf{t}(\mathbf{die}\,\Gamma)$.

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\Gamma$

$$\begin{array}{ll} & \text{true} \\ = & \quad \{\ xa \in \mathbf{t}(\mathbf{die}\,\Gamma) \text{ and property 5.15(ii)}\ \} \\ & x \in \mathbf{t}(\mathbf{die}\,\Gamma) \wedge xa \in \mathbf{t}(\mathbf{die}\,\Gamma) \\ = & \quad \{\ \text{induction hypothesis, using } lx < lt\ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge xa \in \mathbf{t}(\mathbf{die}\,\Gamma) \\ = & \quad \{\ \text{definition 5.14, "}\mathbf{die}\text{", and definition 4.36, "}\mathbf{dse}\text{"}\ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)) \wedge s\mathbf{c}_{\mathbf{io}\,\Gamma} x : \#_a s > \#_a x) \\ = & \quad \{\ \text{property 4.11(i), using } x \in (\mathbf{a}\Gamma)^*\ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)) : s\mathbf{c}_{\mathbf{io}\,\Gamma} xa) \\ \Rightarrow & \quad \{\ \text{definition 5.11, "}\mathbf{DSENTIH}\text{"}\ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge (\mathbf{E}\, s : s \in \mathbf{t}(\mathbf{dse}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma} xa) \\ = & \quad \{\ \text{lemma 4.43}\ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge (\mathbf{E}\, r, s : r \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge s \in \mathbf{t}(\mathbf{dse}\,\Gamma) : r\mathbf{c}_{\mathbf{io}\,\Gamma} s \wedge s\mathbf{c}_{\mathbf{io}\,\Gamma} xa) \\ \Rightarrow & \quad \{\ \text{property 4.10, "transitivity of composability"}\ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge (\mathbf{E}\, r : r \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : r\mathbf{c}_{\mathbf{io}\,\Gamma} xa) \\ = & \quad \{\ \text{property 4.11(i), using } x \in (\mathbf{a}\Gamma)^*\ \} \\ & x \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge (\mathbf{E}\, r : r \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge r\mathbf{c}_{\mathbf{io}\,\Gamma} x : \#_a r > \#_a x) \\ = & \quad \{\ \text{definition 4.36, "}\mathbf{dse}\text{", using } a \in \mathbf{o}\Gamma\ \} \\ & xa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \end{array}$$

case 1: $a \in \mathbf{i}\Gamma$

Suppose there exist trace s and symbol b such that $s \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, $b \in \mathbf{i}\Gamma$, $s\mathbf{c}_{\mathbf{io}\,\Gamma}xa$, and $\#_b s < \#_b xa$. We first prove that $sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$.

true
= { lemma 5.16, using $s \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, $s\mathbf{c}_{\mathbf{io}\,\Gamma}xa$, and $xa \in \mathbf{t}(\mathbf{die}\,\Gamma)$ }
$s \in \mathbf{t}(\mathbf{ptr}\,(\mathbf{DSENTIH}\,\Gamma))$
= { definition 4.36, "**dse**", using $xa \in \mathbf{t}(\mathbf{dse}\,(\mathbf{DSENTIH}\,\Gamma))$, $a \in \mathbf{i}\Gamma$, $s \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, $b \in \mathbf{i}\Gamma$, $s\mathbf{c}_{\mathbf{io}\,\Gamma}xa$, and $\#_b s < \#_b xa$ }
$sb \in \mathbf{t}(\mathbf{ptr}\,(\mathbf{DSENTIH}\,\Gamma))$
⇒ { definition 5.11, "**DSENTIH**" }
$sb \in \mathbf{t}(\mathbf{dse}\,\Gamma)$
= { property 4.5(i) and property 4.11(iii), using $b \in \mathbf{i}\Gamma$ }
$sb \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge s\mathbf{c}_{\mathbf{io}\,\Gamma}sb \wedge \#_b s < \#_b sb$
⇒ { definition 4.36, "**dse**", using $s \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$ and $b \in \mathbf{i}\Gamma$ }
$sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$

We have proven that

$$(\mathbf{A}\,s, b : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge b \in \mathbf{i}\Gamma \wedge s\mathbf{c}_{\mathbf{io}\,\Gamma}xa \wedge \#_b s < \#_b xa : sb \in \mathbf{t}(\mathbf{ptr}\,\Gamma)).(*)$$

true
= { induction hypothesis, using $lx < lt$ }
$x \in \mathbf{t}(\mathbf{dse}\,\Gamma)$
= { definition 4.36, "**dse**", using $a \in \mathbf{i}\Gamma$ and $(*)$ }
$xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$

From $t = xa$ we conclude that $t \in \mathbf{t}(\mathbf{dse}\,\Gamma)$.

end of lemma

lemma 5.19

For di-initializable component Γ,

$$\mathbf{die}\,\Gamma \subseteq \mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma).$$

proof

Given di-initializable component Γ. Let trace t be such that $t \in \mathbf{t}(\mathbf{die}\,\Gamma)$.

Suppose $t \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$

$$\begin{array}{ll} & \text{true} \\ = & \{\ \text{lemma 5.17, using } t \in \mathbf{t}(\mathbf{die}\,\Gamma)\ \} \\ & t \in \mathbf{t}(\mathbf{dse}\,\Gamma) \\ = & \{\ \text{definition 5.11, "}\mathbf{DSENTIH}\text{", using } t \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))\ \} \\ & (\mathbf{E}\,u : u \in (\mathbf{o}\Gamma)^* : tu \in \mathbf{tih}\,\Gamma) \end{array}$$

Given such a trace u; hence, $u \in (\mathbf{o}\Gamma)^*$ and $tu \in \mathbf{tih}\,\Gamma$. From property 1.35 we conclude that there exist a trace x and a symbol a such that $x \in (\mathbf{a}\Gamma)^*$, $a \in \mathbf{i}\Gamma$, $xa\,\mathbf{prefix}\,tu$, $x \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$, and $xa \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$. Given such x and a. Since $a \in \mathbf{i}\Gamma$ and $u \in (\mathbf{o}\Gamma)^*$, we infer $xa\,\mathbf{prefix}\,t$. From property 5.15(ii) and $t \in \mathbf{t}(\mathbf{die}\,\Gamma)$, we conclude that $xa \in \mathbf{t}(\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma))$.

$$\begin{array}{ll} & \text{true} \\ = & \{\ \text{property 4.5(i) and property 4.11(iii), using } x \in (\mathbf{a}\Gamma)^* \text{ and } a \in \mathbf{i}\Gamma\ \} \\ & x\,\mathbf{c}_{\mathbf{io}\Gamma}\,xa \wedge \#_a x < \#_a xa \\ \Rightarrow & \{\ \text{definition 4.36, "}\mathbf{dse}\text{", using } xa \in \mathbf{t}(\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma)),\ x \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)),\text{ and } a \in \mathbf{i}\Gamma\ \} \\ & xa \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)) \\ = & \{\ xa \notin \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))\ \} \\ & \text{false} \end{array}$$

This is a contradiction.

Hence, $t \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$.

end of lemma

lemma 5.20

For di-initializable component Γ,

$$(\mathbf{die}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{C}_4 .$$

proof

Given di-initializable component Γ. From lemma 4.51 we derive that $(\mathbf{die}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4$.

Suppose $(\mathbf{E}\,t, a : t \in (\mathbf{a}\Gamma)^* \wedge a \in \mathbf{a}\Gamma : taa \in \mathbf{t}(\mathbf{die}\,\Gamma))$.

Given such a trace t and such a symbol a; hence, $t \in (\mathbf{a}\Gamma)^*$, $a \in \mathbf{a}\Gamma$, and $taa \in \mathbf{t}(\mathbf{die}\,\Gamma)$. We conclude from definition 5.14, "**die**", that $taa \in \mathbf{t}(\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma))$.

$$\begin{array}{ll} & \text{true} \\ = & \{\ \text{lemma 5.19, using } taa \in \mathbf{t}(\mathbf{die}\,\Gamma)\ \} \\ & taa \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)) \\ = & \{\ \text{definition 5.11, "}\mathbf{DSENTIH}\text{"}\ \} \\ & taa \in \mathbf{redts}(\mathbf{dse}\,\Gamma, \mathbf{i}\,\Gamma, \mathbf{tih}\,\Gamma) \\ \Rightarrow & \{\ \text{definition 1.34, "}\mathbf{redts}\text{"}\ \} \\ & taa \in \mathbf{t}(\mathbf{dse}\,\Gamma) \wedge taa \notin \mathbf{tih}\,\Gamma \\ = & \{\ \text{definition 5.10, "}\mathbf{tih}\text{"}\ \} \\ & \text{false} \end{array}$$

This is a contradiction.

Hence, $(\mathbf{A}\,t, a : t \in (\mathbf{a}\Gamma)^* \wedge a \in \mathbf{a}\Gamma : taa \notin \mathbf{t}(\mathbf{die}\,\Gamma))$. Using $(\mathbf{die}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{D}_4$ we conclude from theorem 5.2, "$\mathbf{C}_4$", that $(\mathbf{die}\,\Gamma, \mathbf{ab}\Gamma) \in \mathbf{C}_4$.

end of lemma

theorem 5.24 *delay-insensitive enclosure*

For di-initializable component Γ,

$$\mathbf{ptr}(\mathbf{DIE}\,\Gamma) = \mathbf{die}\,\Gamma .$$

proof

Given di-initializable component Γ. Let component Γ' be such that $\mathbf{io}\,\Gamma' = \mathbf{io}\,\Gamma$ and $\mathbf{ptr}\,\Gamma' = \mathbf{die}\,\Gamma$. We first prove that Γ' satisfies (i) through (iv) in definition 5.9, "delay-insensitive enclosure".

From property 5.21 and property 5.22 we infer that $\Gamma\ \mathit{NCIHDS}\ \overline{\Gamma'}$. From property 5.18 we conclude that
$(\mathbf{A}\,a,t : a \in \mathbf{o}\Gamma \wedge ta \in \mathbf{t}(\mathbf{ptr}\,\Gamma') : (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,ta))$. From lemma 5.20 we conclude that $(\mathbf{ptr}\,\Gamma', \mathbf{ab}\Gamma) \in \mathbf{C}_4$.

We now show the maximality of $\mathbf{die}\,\Gamma$ by proving that if a component Δ satisfies (i) through (iv) in definition 5.9, "delay-insensitive enclosure", then $\mathbf{ptr}\,\Delta \subseteq \mathbf{die}\,\Gamma$. Let component Δ be such that $\mathbf{io}\,\Delta = \mathbf{io}\,\Gamma$, $\Gamma\ \mathit{NCIHDS}\ \overline{\Delta}$, $(\mathbf{A}\,a,t : a \in \mathbf{o}\Gamma \wedge ta \in \mathbf{t}(\mathbf{ptr}\,\Delta) : (\mathbf{E}\,s : s \in \mathbf{t}(\mathbf{ptr}\,\Gamma) : s\mathbf{c}_{\mathbf{io}\,\Gamma}\,ta))$, and $(\mathbf{ptr}\,\Delta, \mathbf{ab}\Gamma) \in \mathbf{C}_4$.

Suppose $(\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{ptr}\,\Delta) : t \notin \mathbf{t}(\mathbf{die}\,\Gamma))$

Given such a trace t. From property 2.34(ii) and property 5.15(i) it follows that
$(\mathbf{E}\,x,a : x \in (\mathbf{a}\Gamma)^* \wedge a \in \mathbf{a}\Gamma \wedge xa\,\mathbf{prefix}\,t : x \in \mathbf{t}(\mathbf{die}\,\Gamma) \wedge xa \notin \mathbf{t}(\mathbf{die}\,\Gamma))$. Given such a trace x and such a symbol a.

From definition 4.34, "delay-safe enclosure", we infer that $\mathbf{ptr}\,\Delta \subseteq \mathbf{dse}\,\Gamma$. Hence, $xa \in \mathbf{t}(\mathbf{dse}\,\Gamma)$. From lemma 5.19 we infer that $x \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$. Now, we conclude from definition 5.11, "**DSENTIH**", and definition 1.34, "**redts**", that $a \in \mathbf{i}\Gamma$ and furthermore, using property 5.18, $(\mathbf{E}\,y : y \in (\mathbf{a}\Gamma)^* : x = ya)$ or $(\mathbf{E}\,z,b : z \in (\mathbf{o}\Gamma)^* \wedge b \in \mathbf{o}\Gamma : xazbb \in \mathbf{t}(\mathbf{dse}\,\Gamma))$.

case 0: $(\mathbf{E}\,y : y \in (\mathbf{a}\Gamma)^* : x = ya)$

Given such a trace y. Now, $xa = yaa$. Since $xa \in \mathbf{t}(\mathbf{ptr}\,\Delta)$, we conclude from theorem 5.2, "$\mathbf{C}_4$", that $(\mathbf{ptr}\,\Delta, \mathbf{ab}\Gamma) \notin \mathbf{C}_4$.

case 1: $(\mathbf{E}\,z,b : z \in (\mathbf{o}\Gamma)^* \wedge b \in \mathbf{o}\Gamma : xazbb \in \mathbf{t}(\mathbf{dse}\,\Gamma))$

Given such a trace z and such a symbol b. Using $xazbb \in \mathbf{t}(\mathbf{dse}\,\Gamma)$ we conclude from lemma 4.43 that $(\mathbf{E}\,w : w \in \mathbf{ptr}\,\Gamma : w\mathbf{c}_{\mathbf{io}\,\Gamma}\,xazbb)$. Using $z \in (\mathbf{o}\Gamma)^*$, $b \in \mathbf{o}\Gamma$, $xa \in \mathbf{t}(\mathbf{ptr}\,\Delta)$, and $\Gamma\ \mathit{NCIHDS}\ \overline{\Delta}$, we conclude that $xazbb \in \mathbf{t}(\mathbf{ptr}\,\Delta)$. From theorem 5.2, "$\mathbf{C}_4$", it follows that $(\mathbf{ptr}\,\Delta, \mathbf{ab}\Gamma) \notin \mathbf{C}_4$.

We conclude that $(\mathbf{ptr}\,\Delta, \mathbf{ab}\Gamma) \notin \mathbf{C}_4$. This is a contradiction.

Hence, $\mathbf{ptr}\,\Delta \subseteq \mathbf{die}\,\Gamma$. As a consequence, the maximum in property 4.11(i), exists and $\mathbf{die}\,\Gamma$ is maximal.

end of theorem

lemma 5.33

For di-initializable component Γ,

$$(\mathbf{A}\, t, u : t \in \mathbf{t}(\mathbf{ptr}\,\Gamma) \wedge t\mathbf{c}_{\mathbf{io}\,\Gamma} u \wedge u \in \mathbf{t}(\mathbf{die}\,\Gamma) : t \in \mathbf{t}(\mathbf{die}\,\Gamma)).$$

proof

Given component Γ and trace t such that $: t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$. We distinguish two cases:

case 0: $\neg\,(\mathbf{E}\, u : t\mathbf{c}_{\mathbf{io}\,\Gamma} u : u \in \mathbf{t}(\mathbf{die}\,\Gamma))$

Since the universal quantification over an empty domain holds, we are done.

case 1: $(\mathbf{E}\, u : t\mathbf{c}_{\mathbf{io}\,\Gamma} u : u \in \mathbf{t}(\mathbf{die}\,\Gamma))$

Given such a trace u; hence, $t\mathbf{c}_{\mathbf{io}\,\Gamma} u$ and $u \in \mathbf{t}(\mathbf{die}\,\Gamma)$.

true
= { lemma 5.17, using $: t \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, $t\mathbf{c}_{\mathbf{io}\,\Gamma} u$, and $u \in \mathbf{t}(\mathbf{die}\,\Gamma)$ }
$t \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma))$
= { definition 5.14, "die", using $u \in \mathbf{t}(\mathbf{die}\,\Gamma)$ }
$t \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)) \wedge u \in \mathbf{t}(\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma))$
⇒ { lemma 4.49 }
$t \in \mathbf{t}(\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma))$
= { definition 5.14, "die" }
$t \in \mathbf{t}(\mathbf{die}\,\Gamma)$

end of lemma

theorem 5.34

For di-initializable components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$, $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$, and $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$,

$$\mathbf{die}\,\Gamma = \mathbf{die}\,\Delta.$$

proof

Given di-initializable components Γ and Δ such that $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$, $\mathbf{cbds}\,\Gamma \subseteq \mathbf{ptr}\,\Delta$, and $\mathbf{ptr}\,\Delta \subseteq \mathbf{ptr}\,\Gamma$. From theorem 4.80 we infer that $\mathbf{dse}\,\Gamma = \mathbf{dse}\,\Delta$.

true

$=$ { definition 5.10, "**tih**", using $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$ and $\mathbf{dse}\,\Gamma = \mathbf{dse}\,\Delta$ }

$\mathbf{tih}\,\Gamma = \mathbf{tih}\,\Delta$

$\Rightarrow$ { definition 5.11, "**DSENTIH**", using $\mathbf{io}\,\Gamma = \mathbf{io}\,\Delta$ and $\mathbf{dse}\,\Gamma = \mathbf{dse}\,\Delta$ }

$\mathbf{DSENTIH}\,\Gamma = \mathbf{DSENTIH}\,\Delta$

$\Rightarrow$ { definition 5.14, "**die**" }

$\mathbf{die}\,\Gamma = \mathbf{die}\,\Delta$

end of theorem

theorem 5.44

For di-initializable component Γ,

$$\mathbf{cbdi}\,\Gamma = \mathbf{cbds}\,\Gamma \cap \mathbf{die}\,\Gamma.$$

proof

Given di-initializable component Γ.

$\mathbf{cbdi}\,\Gamma$
$=$ { property 5.43 }
$\mathbf{ptr}\,\Gamma \cap \mathbf{die}\,\Gamma$
$=$ { lemma 5.17 }
$\mathbf{ptr}\,\Gamma \cap \mathbf{dse}\,\Gamma \cap \mathbf{die}\,\Gamma$
$=$ { property 4.72 }
$\mathbf{cbds}\,\Gamma \cap \mathbf{die}\,\Gamma$

end of theorem

theorem 5.48 *maximal communication behavior for delay-insensitive communication*
For di-initializable component Γ,

$$\mathbf{ptr}(\mathbf{CBDI}\,\Gamma) = \mathbf{cbdi}\,\Gamma .$$

proof

Given di-initializable component Γ. From property 5.43 we derive that $\mathbf{cbdi}\,\Gamma \subseteq \mathbf{ptr}\,\Gamma$. From definition 5.41, "**cbdi**", we derive that $(\mathbf{A}\,a,s : a \in \mathbf{i}\Gamma \wedge sa \in \mathbf{t}(\mathbf{cbdi}\,\Gamma) : (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{die}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\,\Gamma}t))$.

We now have to prove that $\mathbf{cbdi}\,\Gamma$ is maximal. Let trace s and symbol a be such that $s \in \mathbf{t}(\mathbf{cbdi}\,\Gamma)$, $sa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)$, and $a \in \mathbf{i}\Gamma \Rightarrow (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{die}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\,\Gamma}t)$. We prove that $sa \in \mathbf{t}(\mathbf{cbdi}\,\Gamma)$.

Since $\mathbf{a}\Gamma$ is bipartitioned into $\mathbf{o}\Gamma$ and $\mathbf{i}\Gamma$, we distinguish:

case 0: $a \in \mathbf{o}\Gamma$

$$\begin{array}{ll} & \text{true} \\ = & \quad \{\ \text{property 5.43, using } s \in \mathbf{t}(\mathbf{cbdi}\,\Gamma)\ \} \\ & s \in \mathbf{t}(\mathbf{die}\,\Gamma) \\ = & \quad \{\ \text{property 4.6(i), using } a \in \mathbf{o}\Gamma\ \} \\ & sa\mathbf{c}_{\mathbf{io}\,\Gamma}s \wedge s \in \mathbf{t}(\mathbf{die}\,\Gamma) \\ \Rightarrow & \quad \{\ \text{lemma 5.16, using } sa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)\ \} \\ & sa\mathbf{c}_{\mathbf{io}\,\Gamma}s \wedge sa \in \mathbf{t}(\mathbf{ptr}(\mathbf{DSENTIH}\,\Gamma)) \\ \Rightarrow & \quad \{\ \text{definition 4.36, "dse", using } a \in \mathbf{o}\Gamma\ \} \\ & sa \in \mathbf{t}(\mathbf{dse}(\mathbf{DSENTIH}\,\Gamma)) \\ = & \quad \{\ \text{definition 5.14, "die",}\ \} \\ & sa \in \mathbf{t}(\mathbf{die}\,\Gamma) \\ = & \quad \{\ \text{property 5.43, using } sa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)\ \} \\ & sa \in \mathbf{t}(\mathbf{cbdi}\,\Gamma) \end{array}$$

case 1: $a \in \mathbf{i}\Gamma$

$$\begin{array}{ll} & \text{true} \\ = & \quad \{\ (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{die}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\,\Gamma}t),\ \text{since } a \in \mathbf{i}\Gamma\ \} \\ & (\mathbf{E}\,t : t \in \mathbf{t}(\mathbf{die}\,\Gamma) : sa\mathbf{c}_{\mathbf{io}\,\Gamma}t) \\ = & \quad \{\ \text{definition 5.41, "cbdi", using } sa \in \mathbf{t}(\mathbf{ptr}\,\Gamma)\ \} \\ & sa \in \mathbf{t}(\mathbf{cbdi}\,\Gamma) \end{array}$$

end of theorem

References

[Barney 85]
Clifford Barney
Logic Designers Toss Out the Clock
Electronics, (December 9, 1985), pp. 42-45

[Bisseling – Eemers – Kamps – Peeters 90]
Hans Bisseling, Henk Eemers, Michiel Kamps, and Ad Peeters
Designing Delay-Insensitive Circuits
Eindhoven: University of Technology, September 1990
(Instituut Vervolgopleidingen; Final report of the postgraduate programme Software Technology)
(ISBN: 90-5282-076-7)

[Black 86]
David L. Black
On the Existence of Delay-Insensitive Fair Arbiters: Trace Theory and its Limitations
Distributed Computing, **1**, No. 4 (October 1986), pp. 205-225

[Brzozowski – Ebergen 89]
J.A. Brzozowski and J.C. Ebergen
Recent Developments in the Design of Asynchronous Circuits
Waterloo, Canada: University of Waterloo, May 1989
(Computer Science Department; Research Report CS-89-18)
(Presented at the Seventh International Conference "Fundamentals of Computation Theory", FCT'89, Hungary, August 1989)

[Brzozowski – Seger 89]
J.A. Brzozowski and C.-J. Seger
A Unified Framework for Race Analysis of Asynchronous Networks
Journal of the ACM, **36**, No. 1 (January 1989), pp. 20-45

[Burstyn86]
David Burstyn
State-Graph Based Verification Techniques for Delay-Insensitive Modules
St. Louis: Washington University, April 1986
(Institute for Biomedical Computing, Computer Systems Laboratory; technical memorandum 318)

[Chaney86]
T.J.Chaney
A Comprehensive Bibliography on Synchronizers and Arbiters
St. Louis: Washington University, December 1986
(Institute for Biomedical Computing, Computer Systems Laboratory; technical memorandum 306C)

[Chaney – Molnar73]
T.J.Chaney and C.E. Molnar
Anomalous Behavior of Synchronizer and Arbiter Circuit
IEEE Transactions on Computers, **C-22** (April 1973), pp. 421-422

[Chen – Udding – Verhoeff89]
Wei Chen, Jan Tijmen Udding, and Tom Verhoeff
Networks of Communicating Processes and Their (De-)Composition
in: *Mathematics of Program Construction*; pp. 174-196
Berlin: Springer, 1989
(Lecture Notes in Computer Science, Vol. 375)
(Proceedings International Conference: 375th Anniversary of the Groningen University, June 1989; ed. by J.L.A. van de Snepscheut)

[Clark – Molnar74]
W.A. Clark, and C.E. Molnar
Macromodular Computer Systems
in: *Computers in Biomedical Research*, Vol. IV; ed. by R.W. Stacey and B.D. Waxman, pp. 45-85
New York: Academic Press, 1974

[Cox85]
J.R. Cox
Arbiter Specification and Fair Behavior
St. Louis: Washington University, November 1985
(Department of Computer Science; internal memorandum)

[Dally – Seitz 86]
William J. Dally and Charles L. Seitz
The Torus Routing Chip
Distributed Computing, **1**, No. 4 (October 1986), pp. 187-196

[de Graaff 86]
P.J. de Graaff
A Design Method for Self-Timed Circuits
Eindhoven: University of Technology, July 1986
(Department of Mathematics and Computing Science; Master's thesis)

[Dill 88]
David L. Dill
Automatic Hierarchical Verification of Speed-Independent Circuits
Pittsburgh: Carnegie Mellon University, February 1988
(Computer Science Department; memorandum CMU-CS-88-119)

[Dill – Clarke 85]
David L. Dill and Edmund M. Clarke
Automatic Verification of Asynchronous Circuits Using Temporal Logic
in: *1985 Chapel Hill Conference on Very Large Scale Integration*; ed. by Henry Fuchs, pp. 127-143
Rockville: Computer Science Press, 1985

[Dijkstra – Feijen 88]
Edsger W. Dijkstra and W.H.J. Feijen
A Method of Programming
Amsterdam: Addison-Wesley, 1988
(Translated by Joke Sterringa; original publication: Een Methode van Programmeren; 's-Gravenhage: Academic Service, 1984)
(ISBN: 0-201-17536-3)

[Ebergen 87]
Jo C. Ebergen
Translating Programs into Delay-Insensitive Circuits
Eindhoven: University of Technology, October 1987
(Department of Mathematics and Computing Science; doctoral dissertation)

[Ebergen 88]
Jo C. Ebergen
A Formal Approach to Designing Delay-Insensitive VLSI Circuits
Eindhoven: University of Technology, February 1988
(Department of Mathematics and Computing Science; Computing Science Notes 88/10)

[Fang 87]
Ting-Pien Fang
On Decomposition by Factoring
St. Louis: Washington University, 1987
(Institute for Biomedical Computing; internal memorandum)

[Fang – Molnar 83]
T.P. Fang and C.E. Molnar
Synthesis of Reliable Speed-Independent Circuit Modules: II. Circuit and Delay Conditions to Ensure Operation Free of Problems from Races and Hazards
St. Louis: Washington University, December 1983
(Institute for Biomedical Computing, Computer Systems Laboratory; technical memorandum 298)

[Greenstreet – Williams – Staunstrup 88]
M. Greenstreet, T. Williams, and J. Staunstrup
Self-Timed Iteration
in: *VLSI 87, VLSI Design of Digital Systems*; pp. 309-322
Amsterdam: North Holland, 1988
(Proceedings of the IFIP International Conference on Very Large Scale Integration, Vancouver; ed. by Carlo H. Séquin)

[Hoare 85]
C.A.R. Hoare
Communicating Sequential Processes
Englewood Cliffs: Prentice-Hall, 1985
(Prentice-Hall International Series in Computer Science)

[Huffman54]
D.A. Huffman
The Synthesis of Sequential Switching Machines
J. Franklin Institute, **257**, No. 3 (March 1954), pp.161-190, and No. 4 (April 1954), pp.275-305
(Reprinted in E.F. Moore, Sequential Machines: Selected Papers; Reading: Addison-Wesley, 1964)

[Hurtado75]
Marco Hurtado
Structure and Performance of Asymptotically Bistable Dynamical Systems
St. Louis: Washington University, May 1975
(Sever Institute; doctoral dissertation)

[Josephs – Hoare – Jifeng89]
Mark B. Josephs, C.A.R. Hoare, and He Jifeng
A Theory of Asynchronous Processes
Oxford: Oxford University, 1989
(Oxford University Computing Laboratory, Programming Research Group)

[Josephs – Udding89]
Mark B. Josephs and Jan Tijmen Udding
An Algebra for Delay-insensitive Circuits
St. Louis: Washington University, 1989
(Department of Computer Science; technical report WUCS-89-54)

[Josephs – Udding90]
Mark B. Josephs and Jan Tijmen Udding
Delay-insensitive Circuits: an Algebra Approach to their Design
in: *CONCUR'90 Theories of Concurrency: Unifications and Extensions*; ed. by J.C.M. Baeten and J.W. Klop, pp. 342-366
Berlin: Springer, 1990
(Lecture Notes in Computer Science, Vol. 458)

[Kaldewaij86]
Anne Kaldewaij
A Formalism for Concurrent Processes
Eindhoven: University of Technology, May 1986
(Department of Mathematics and Computing Science; doctoral dissertation)

[Keller74]
Robert M. Keller
Towards a Theory of Universal Speed-Independent Modules
IEEE Transactions on Computers, **C-23**, No. 1 (January 1974), pp. 21-33

[Keller75]
Robert M. Keller
A Fundamental Theorem of Asynchronous Parallel Computation
in: *Parallel Processing*; pp. 102-112
Berlin: Springer, 1975
(Lecture Notes in Computer Science, Vol. 24)
(Proceedings of the Sagamore Computer Conference)

[Kimura79]
Takayuki Kimura
Behavioral Abstraction of Communicating Sequential Processes
St. Louis: Washington University, January 1979
(Department of Computer Science; internal memorandum)

[Kleemann – Cantoni87]
Lindsay Kleemann and Antonio Cantoni
Metastable Behavior in Digital Systems
IEEE Design & Test of Computers, (December 1987) pp. 4-19

[Langenberg92]
J.C.F.M. Langenberg
Designing Processes using Invariants,
Eindhoven: University of Technology, June 1992
(Department of Mathematics and Computing Science; Master's thesis)

[Mago85]
G.A. Mago
Making Parallel Computation Simple : the FFP Machine
in: *IEEE Spring Compcon*, 1985, pp. 424-428
(Reprinted in: Benjamin Wak and G.J. Li, Computers for Artificial Intelligence Applications, IEEE Computer Press, 1986, pp. 329-338)

[Martin85a]
Alain J. Martin
The Probe: An Addition to Communication Primitives
Information Processing Letters, **20** (1985), pp. 125-130, and **21** (1985), p. 107

[Martin85b]
Alain J. Martin
The Design of a Self-Timed Circuit for Distributed Mutual Exclusion
in: *1985 Chapel Hill Conference on Very Large Scale Integration*; ed. by Henry Fuchs, pp. 245-260
Rockville: Computer Science Press, 1985

[Martin86]
Alain J. Martin
Compiling Communicating Processes into Delay-Insensitive VLSI Circuits
Distributed Computing, **1**, No. 4 (October 1986), pp. 226-234

[Martin87]
Alain J. Martin
A Synthesis Method for Self-timed VLSI Circuits
in: *Proceedings ICCD '87*, pp. 224-229
IEEE Computer Society Press, 1987
(IEEE International Conference on Computer Design: VLSI in Computers & Processors)

[Martin90]
Alain J. Martin
Programming in VLSI : From Communicating Processes to Delay-Insensitive Circuits
in: *Developments in Concurrency and Communication*; ed. by C.A.R. Hoare
Amsterdam: Addison-Wesley, 1990
(Proceedings UT Year of Programming Institute on Concurrent Programming 1988, Austin)
(UT Year of Programming series; ed. by Hamilton Richards Jr.)

[Martin – Burns – Lee – Borkovic – Hazewindus 89]
Alain J. Martin, Steven M. Burns, T.K. Lee, Drazen Borkovic, and Pieter J. Hazewindus
The Design of an Asynchronous Microprocessor
in: *Advanced Research in VLSI: proceedings of the Decennial Caltech Conference on VLSI*; ed. by Charles L. Seitz, pp. 351-373
Massachusetts: Institute of Technology, 1989

[Mazurkiewicz 85]
A. Mazurkiewicz
Semantics of Concurrent Systems: A Modular Fixed Point Trace Approach
in: *Advances in Petri nets 1984*; ed. by G. Rozenberg, pp. 353-375
Berlin: Springer, 1985
(Springer Lecture Notes in Computer Science 188)

[Miller 65]
R.E. Miller
Speed Independent Switching Circuit Theory
chapter 10 in: *Switching Theory. Vol. II: Sequential Circuits and Machines*
New York: John Wiley, 1965

[Moll 85]
Erik Moll
Limitations of Delay-Insensitive Trace Structures
Eindhoven: University of Technology, May 1985
(Department of Mathematics and Computing Science; Master's thesis)

[Molnar 85]
Charles E. Molnar
Specifications vs. Descriptions
St. Louis: Washington University, November 1985
(Institute for Biomedical Computing; internal memorandum)

[Molnar 86]
Charles E. Molnar
Introduction to Asynchronous Systems
in: *Proceedings New Frontiers in Computer Science Conference*; pp. 83-93
Santa Monica: Citicorp/TTI, March 1986

[Molnar92]
Charles E. Molnar
Private communication.

[Molnar – Fang81]
C.E. Molnar and T.P. Fang
An Asynchronous System Design Methodology
St. Louis: Washington University, June 1981
(Institute for Biomedical Computing, Computer Systems Laboratory; technical memorandum 287)

[Molnar – Fang83]
C.E. Molnar and T.P. Fang
Synthesis of Reliable Speed-Independent Circuit Modules : I. General Method for Specification of Module-Environment Interaction and Derivation of a Circuit Realization
St. Louis: Washington University, 1983
(Institute for Biomedical Computing, Computer Systems Laboratory; technical memorandum 297)

[Molnar – Fang – Rosenberger85]
Charles E. Molnar, Ting-Pien Fang, and Frederick U. Rosenberger
Synthesis of Delay-Insensitive Modules
in: *1985 Chapel Hill Conference on Very Large Scale Integration*; ed. by Henry Fuchs, pp. 67-86
Rockville: Computer Science Press, 1985

[Muller – Bartky59]
David E. Muller and W.S. Bartky
A Theory of Asynchronous Circuits
in: *Proceedings of an International Symposium on the Theory of Switching 2-5 April 1957*; pp. 204-243
Boston: Harvard University Press, 1959

[Rem 85]
Martin Rem
Concurrent Computations and VLSI Circuits
in: *Control Flow and Data Flow: Concepts of Distributed Programming*; pp. 399-437
Berlin: Springer, 1985
(Proceedings of the NATO Advanced Study Institute, 1984; ed. by M. Broy)
(NATO ASI series. Series F: Computer and Systems Sciences; Vol. 14)

[Rem 91]
Martin Rem
The Nature of Delay-Insensitive Computing
in: *IV Higher Order Workshop Banff 1990*; ed. by Graham Birtwistle, pp. 105-122
Berlin: Springer-Verlag, 1991

[Rem – van de Snepscheut – Udding 83]
Martin Rem, Jan L.A. van de Snepscheut, and Jan Tijmen Udding
Trace Theory and the Definition of Hierarchical Components
in: *Proceedings of the third Caltech Conference on Very Large Scale Integration*; ed. by Randal Bryant, pp. 225-240
Rockville: Computer Science Press, 1983

[Rosenberger 69]
Fred U. Rosenberger
Control of Concurrent Operations in Asynchronous Digital Processes
St. Louis: Washington University, May 1969
(D. Sc. thesis. External publication FR312)

[Rosenberger – Molnar – Chaney – Fang 88]
F.U. Rosenberger, C.E. Molnar, T.J. Chaney, and T.P. Fang
Q-modules: Internally Clocked Delay-Insensitive Modules
IEEE Transactions on Computers, **C-37** (1988), pp. 1005-1018

[Schols 85]
Huub M.J.L. Schols
A Formalisation of the Foam Rubber Wrapper Principle,
Eindhoven: University of Technology, February 1985
(Department of Mathematics and Computing Science; Master's thesis)

[Schols86]
Huub M.J.L. Schols
Partial Delay-Insensitivity in Trace Theory
Eindhoven: University of Technology, February 1986
(Department of Mathematics and Computing Science; internal memorandum)

[Schols88]
Huub M.J.L. Schols
Notes on Delay-Insensitive Communication
Eindhoven: University of Technology, March 1988
(Department of Mathematics and Computing Science; Computing Science Notes 88/06)

[Seger88]
Carl-Johan Henry Seger
Models and Algorithms for Race Analysis in Asynchronous Networks
Waterloo: University of Waterloo, 1988
(Faculty of Mathematics; research report CS-88-22)

[Seitz79]
Charles L. Seitz
Self-Timed VLSI Systems
in: *Proceedings of the (first) Caltech Conference on Very Large Scale Integration*; ed. by Charles L. Seitz, pp. 345-356
Pasadena: California Institute of Technology, 1979
(Department of Computer Science)

[Seitz80]
Charles L. Seitz
System Timing
chapter 7 in: Carver A. Mead and Lynn A. Conway, *Introduction to VLSI Systems*
Reading: Addison-Wesley, 1980

[Seitz85]
Charles L. Seitz
The Cosmic Cube
Communications of the ACM, **28**, No. 1 (January 1985), pp. 22-33

[Siccama86]
A. Siccama
Minimal Characterisations of Directed Trace Structures
Eindhoven: University of Technology, May 1986
(Department of Mathematics and Computing Science; Master's thesis)

[Sutherland89]
Ivan E. Sutherland
Micropipelines
Communications of the ACM, **32**, No. 6 (June 1989), pp. 720-738
(Turing Award Lecture)

[Udding84]
Jan Tijmen Udding
Classification and Composition of Delay-Insensitive Circuits
Eindhoven: University of Technology, September 1984
(Department of Mathematics and Computing Science; doctoral dissertation)

[Udding85]
Jan Tijmen Udding
On the Non-Existence of Delay-Insensitive Fair Arbiters
St. Louis: Washington University, January 1985
(Institute for Biomedical Computing; internal memorandum JTU14)

[Unger69]
Stephen H. Unger
Asynchronous Sequential Switching Circuits
New York: John Wiley, 1969

[Unger80]
Stephen H. Unger
Equivalence of Synthesis Problems for Arbiters, Synchronizers, and Inertial Delays
in: *Report on the Workshop on Self-Timed Systems*, July 8-12, 1979; ed. by Randal E. Bryant, pp. 8-10
Cambridge, Massachusetts: Massachusetts Institute of Technology, May 1980
(Laboratory for Computer Science; technical memorandum MIT/LCS/TM-166)

[van Berkel92]
C.H. (Kees) van Berkel
Handshake circuits: an intermediary between communicating processes and VLSI
Eindhoven: University of Technology, May 1992
(Department of Mathematics and Computing Science; doctoral dissertation)

[van der Heijden – Teunissen89]
P.J. van der Heijden and R.A.P. Teunissen
The computation of the delay-insensitive enclosure without transmission interference hazard
Eindhoven: University of Technology, August 1989
(Department of Mathematics and Computing Science; Master's thesis)

[van der Veeken87]
R.A.P.M. van der Veeken
A Method for Decomposing Delay-Insensitive Circuits
Eindhoven: University of Technology, October 1987
(Department of Mathematics and Computing Science; Master's thesis)

[van de Snepscheut85]
Jan L.A. van de Snepscheut
Trace Theory and VLSI Design
Berlin: Springer, 1985
(Springer Lecture Notes in Computer Science 200)

[Van Horn86]
Kevin S. Van Horn
An Approach to Concurrent Semantics Using Complete Traces
Pasadena: California Institute of Technology, December 1986
(Department of Computer Science; memorandum 5236:TR:86)

[Verhoeff85]
Tom Verhoeff
Notes on Delay-Insensitivity
Eindhoven: University of Technology, August 1985
(Department of Mathematics and Computing Science; Master's thesis)

[Verhoeff88]
Tom Verhoeff
Delay-Insensitive Codes – An Overview
Distributed Computing, **3**, No. 1 (December 1988), pp. 1-8

[Verhoeff89]
Tom Verhoeff
Notes of the Directed Specifications Group, DSG08
Eindhoven: University of Technology, January 17, 1989
(Department of Mathematics and Computing Science; internal memorandum)

[Verhoeff – Schols85]
Tom Verhoeff and Huub M.J.L. Schols
Delay-Insensitive Directed Trace Structures Satisfy the Foam Rubber Wrapper Postulate
Eindhoven: University of Technology, August 1985
(Department of Mathematics and Computing Science; Computing Science Notes 85/04)

[Yakovlev85]
Alexandre Yakovlev
Designing Self-Timed Systems
VLSI Systems Design, (September 1985), pp. 70-90

[Yoeli87]
Michael Yoeli
Specification and Verification of Asynchronous Circuits Using Marked Graphs
in: *Concurrency and Nets, Advances in Petri Nets*; ed. by H.J. Genrich and G. Rozenberg
Berlin: Springer, 1987

Glossary of symbols and operators

$\varnothing$	empty set
\$	end of trace marker in (general) composability diagram
$\sqsubset_\phi$	strict partial order of comminstorder (*or:* commsigorder) ϕ
$\overline{\Gamma}$	reflection of component Γ
$\overline{\Phi}$	reflection of iodir Φ
A^*	Kleene-closure of set A
$\neg x$	negation: not x
$\#_a t$	the number of occurrences of symbol a in trace t
$x=y$	equality: x equals y
$x\neq y$	inequality: x differs from y
$x \wedge y$	conjunction: x and y
$x \vee y$	disjunction: x or y
$x \Rightarrow y$	implication: x implies y
$[\varepsilon]$	initial state
$[t]$	state to which trace t corresponds
$<a!>$	allowed transition a (possibly leaving a lazy state)
$<A, S>$	trace structure with alphabet A and trace set S
(α, i, β)	i-th commsig with output commport α and input commport β
$\{a, b\}$	set with elements a and b
$\{l : R : e\}$	quantification denoting a set
$S\restriction A$	projection of trace set (*or:* trace structure) S on alphabet A
$s\restriction A$	projection of trace s on alphabet A
$\phi\restriction \Lambda$	restriction of comminstorder (*or:* commsigorder) ϕ to initial set of comminsts (*or:* commsigs) Λ
$a \in A$	member: a is an element of set A
$a \notin A$	nonmember: a is not an element of set A
$A \subset B$	set A is a proper subset of set B
$T \subset U$	trace structure T is properly included in trace structure U
$A \subseteq B$	set A is a subset of set B
$T \subseteq U$	trace structure T is included in trace structure U
$P \cap Q$	intersection of sets (*or:* trace structures) P and Q
$P \cup Q$	union of sets (*or:* trace structures) P and Q

$A \oplus B$	alphbip consisting of disjoint alphabets A and B
$A \div B$	symmetric set difference of sets A and B
$A \backslash B$	asymmetric set difference: A minus B
$tc_F u$	trace t is composable under iobip F with trace u
$tg_F u$	trace t is generally composable under iobip F with trace u
Λ_ϕ	set of comminsts (*or:* commsigs) of comminstorder (*or:* commsigorder) ϕ
Ξ', Ξ''	disjoint sets of input commports of opdir Ξ
Φ^i	set of input commports of iodir Φ
Φ^o	set of output commports of iodir Φ
Ψ	set of all commports
Ψ^i	set of all input commports
Ψ^i_Δ	set of input commports of module Δ
Ψ_{ic}	set of all indirectly connected commports
Ψ^o	set of all output commports
Ψ^o_Δ	set of output commports of module Δ
Ω	universe: set of all symbols used in trace theory
ε	empty trace
$(\mathbf{A}\, l : R : E)$	universal quantification
$\mathbf{C}_4$	class of "trace structure" – alphbip pairs related to delay-insensitivity
CB Δ	communication behavior of module Δ
CBDI Γ	component: maximal communication behavior of Γ
CBDS Γ	component: maximal communication behavior of Γ
CBNTIHI$_A$ Γ	component: reduction of Γ by **tihi**$_A$ Γ
CM Π	communication in interconnection Π
Γ ***COMPNCIH***$_I$ Δ	composite of Γ and Δ without computation interference hazard
Γ ***COMPNCTIH***$_I$ Δ	composite of Γ and Δ without computation and transmission interference hazard
D	alphabet associated with the directly connected commports
$\mathbf{D}_4$	class of "trace structure" – alphbip pairs related to delay-safety
DIE Γ	component: delay-insensitive enclosure of component Γ
DSC Θ	channel: delay-safe closure of channel Θ
DSE Γ	component: delay-safe enclosure of component Γ
$(\mathbf{E}\, l : R : E)$	existential quantification
I	state mark: interference indicator
I	alphabet associated with the indirectly connected commports
IO Δ	iodir of module Δ
L	lazy state mark

α ***MATCH*** β	output commport α is connected to input commport β
$\mathbb{N}$	Set of natural numbers (including *0*)
$\mathbb{N}^+$	Set of positive natural numbers
Γ ***NICIH***$_I$ Δ	components Γ and Δ have no computation interference hazard
Γ ***NICTIH***$_I$ Δ	Γ and Δ have computation nor transmission interference hazard
OP Π	opdir of interconnection Π
REDOC ϕ	comminstorder: reduction of comminstorder ϕ
aΓ, **a**Θ, **a**Ξ, **a**Φ	alphabet of component Γ, channel Θ, opdir Ξ, or iodir Φ
a*D*, **a***F*, **a***T*	alphabet of alphbip *D*, iobip *F*, or trace structure *T*
abΓ, **ab**Θ, **ab**Ξ	alphbip of component Γ, channel Θ, or opdir Ξ
T **b** *U*	blend of trace structures *T* and *U*
bag *s*	bag of trace *s*
cbdiΓ	trace structure of component **CBDI** Γ
cbdsΓ	trace structure of component **CBDS** Γ
cihi$_I$ (Γ, Δ)	trace set associated with computation interference hazard at Δ
dieΓ	trace structure of component **DIE** Γ
dscΘ	mathematical closure within $\mathbf{D}_4$ of channel (*or:* component) Θ
dseΓ	trace structure of component **DSE** Γ
Γ ***extcomncih***$_I$ Δ	trace structure of component Γ ***COMPNCIH***$_I$ Δ
extinp(Γ, Δ)	alphabet associated with the external inputs of the composite
extoutp(Γ, Δ)	alphabet associated with the external outputs of the composite
iΓ, **i**Φ, **i***F*	input alphabet of component Γ, iodir Φ, or iobip *F*
ioΓ, **io**Φ	iobip of component Γ, or iodir Φ
ls	length of trace *s*
oΓ, **o**Φ, **o***F*	output alphabet of component Γ, iodir Φ, or iobip *F*
opa(*a*, *D*)	alphabet: part of alphbip *D* that does not contain symbol *a*
pref *S*	prefix-closure of trace set (*or:* trace structure) *S*
s **prefix** *t*	trace *s* is a prefix of trace *t*
ptrΓ	trace structure: communication behavior of component Γ
ptrΘ	trace structure: communication of channel Θ
redts(*T*, *A*, *S*)	trace structure (reduction of trace structure *T*)
spa(*a*, *D*)	alphabet: part of alphbip *D* that contains symbol *a*
t*T*	trace set of trace structure *T*
tihi$_A$ Γ	trace set associated with transmission interference hazard
Γ ***totcom***$_I$ Δ	trace structure: combination of the trace structures of Γ and Δ when these are composed under *I*
Γ ***totcomncih***$_I$ Δ	trace structure: reduction of Γ ***totcom***$_I$ Δ
T **w** *U*	weave of trace structures *T* and *U*

Subject index

Summary

In this monograph we study delay-insensitive communication. Communication is called delay-insensitive if it is delay-safe and it has absence of transmission interference hazard (no two signals along the same wire may interfere). Communication is called delay-safe if its correctness does not depend on the values of the delays in wires nor on the reaction times of mechanisms. Notice that the communication may depend on these delays or reaction times, as long as the correctness of it remains unchanged. The formalization of delay-safety is based on our causality notion: "no signal is received before it has been sent". There exist various reasons why one may be interested in delay-safe communication, e.g. scaling (the delays in the wires tend to increase relatively to the delays in the switching elements) and metastability (the reaction time of some mechanisms is unbounded).

We introduce our Communication Model as a formal abstraction of 'the underlying physics'. Modules model the physical mechanisms. The terminals of the mechanisms are modeled by commports. We define components as equivalence classes of modules. We distinguish directly and indirectly connected commports; this leads to components that have a direct, an indirect, or (in general) a mixed connection. The correctness concern "absence of computation interference hazard" (a component accepts every input that it may receive) plays a central role in this monograph. We present a technique to transform other correctness concerns into absence of computation interference hazard.

We distinguish between the communication behavior of components and the communication of a channel between them. We present the limitations of delay-safe communication and of delay-insensitive communication. Furthermore, given the correctness concern absence of computation interference hazard (and possibly also absence of transmission interference hazard) we define composition of components that have a mixed connection. We give necessary and sufficient conditions for the existence of the compositions under the given correctness concern(s). We address factorization, which is a form of decomposition in which the specification and a (desired) part of the solution of this specification are given. Since our two correctness concerns are symmetric w.r.t. the specification and all parts of the solution, factorization is equal to composition.

Samenvatting

In dit proefschrift bestuderen we vertragingsongevoelige communicatie. We noemen de communicatie vertragingsongevoelig als de communicatie "vertragingsveilig" is en als er geen gevaar bestaat voor transmissie interferentie (geen tweetal signalen op dezelfde verbindingsdraad kunnen met elkaar interfereren). In vertragingsongevoelige communicatie is de correctheid van de communicatie niet afhankelijk van de vertragingen in verbindingsdraden, noch van de snelheid waarmee een mechanisme reageert op signalen; de communicatie zelf mag wel hiervan afhangen. De formele definitie van vertragingsveiligheid is gebaseerd op ons oorzakelijkheidsbegrip: "geen signaal wordt ontvangen voordat het verstuurd is". Er zijn verscheidene redenen waarom we interesse hebben in vertragingsveilige communicatie, bijv. schaalverkleining (vertragingen in verbindingsdraden hebben de neiging relatief toe te nemen in vergelijking met schakeltijden van transistoren) en meta-stabiliteit (de reactietijden van sommige mechanismen zijn niet naar boven begrensd).

Ons communicatiemodel is een formele abstractie van 'de onderliggende fysische begrippen'. Mechanismen worden gemodelleerd door modulen. De communicatie-poorten van deze modulen modelleren de terminals van de mechanismen. Componenten zijn equivalentieklassen van modulen. We maken onderscheid tussen directe en indirecte verbindingen van communicatie-poorten; op deze manier onderscheiden we directe, indirecte en gemengde verbindingen van componenten. Het correctheidscriterium "geen gevaar voor interferentie van inputs met de lopende berekening" (een component accepteert elke input die hij ontvangt) loopt als een rode draad door dit proefschrift. Er wordt een methode gepresenteerd om andere correctheidscriteria te transformeren tot "geen gevaar voor interferentie van inputs met de lopende berekening".

We maken onderscheid tussen het communicatiegedrag van componenten en de communicatie in een kanaal tussen deze componenten. De uiterste grenzen van vertragingsveilige en vertragingsongevoelige communicatie worden aangegeven. Gegeven het correctheidscriterium "geen gevaar voor interferentie van inputs met de lopende berekening" (en eventueel ook "geen gevaar voor transmissie interferentie") definiëren we samenstelling van componenten die een gemengde verbinding hebben. We geven nodig en voldoende voorwaarden voor het bestaan van de samenstellingen van componenten gegeven de correctheidscriteria. Ook

wordt aandacht besteed aan de uitsplitsing van componenten; uitsplitsing is een vorm van ontbinding, waarbij de te ontbinden component en een (gewenst) deel van het resultaat van de ontbinding gegeven zijn. Omdat onze twee correctheidscriteria symmetrisch zijn met betrekking tot de uit te splitsen component en alle delen van het resultaat van de uitsplitsing, komt uitsplitsing neer op samenstelling.

Curriculum vitae

- I was born on August 4, 1955 in Amby, the Netherlands.
- In 1973 I graduated from high-school (viz. Gymnasium-β at "Henric van Veldeke-college) in Maastricht, the Netherlands).
- In 1978 I passed the "kandidaatsexamen" (Bachelor's Degree) in Mathematics at Eindhoven University of Technology in Eindhoven, the Netherlands.
- During 1980 I was a programmer/program-designer with teaching responsabilities at the department of Mechanical Engineering of Eindhoven University of Technology. I developed procedures for programs that solved non-linear stress problems.
- During 1981 and 1982 I performed the military service. After a four-and-a-half month military training, I was a programmer/program-designer in the army in The Hague, the Netherlands. I made a monitor for developing interactive programs.
- In February 1985 I received the Master's Degree in Mathematics from the department of Mathematics and Computing Science of Eindhoven University of Technology. My thesis was in the field of theoretical VLSI-design: "A formalisation of the Foam Rubber Wrapper postulate".
- From March 1985 until February 1987 I was a scientific assistent at the department of Mathematics and Computing Science of Eindhoven University of Technology. I did research in the field of (theoretical) VLSI-design, in particular delay-insensitive circuits.
- Since February 1987 I am a "Universitair Docent" (assistant professor) at the department of Mathematics and Computing Science of Eindhoven University of Technology. I do research and teach in the areas parallelism and VLSI-design. Since 1990 I work part-time at Eindhoven University of Technology.

- From April 1987 until October 1987 I was a visiting research associate at the Institute for Biomedical Computing of Washington University in St.Louis, Missouri, U.S.A. My research in VLSI-design was concentrated on the physical interpretation of formal models.

- In 1990 I was a part-time advisor to Mobius B.V. in Vught, the Netherlands. I gave scientific advise in the areas authentication, encryption, and identification.

- Since January 1991 I work part-time as the technical coordinator of Mobius B.V.

Statements

that go with the Ph.D. Thesis

Delay-insensitive Communication

by

Huub Schols

Eindhoven,
December 9, 1992

0 Delay-safety is not a property of some physical circuits: at the circuit level it is an assumption about delays. In a formal model it can be defined as a property of formal objects.

lit.: – This thesis.

1 Udding's composability operator is suited to be used to model the matching of behaviors of mechanisms that exchange signals in a delay-insensitive way; this can be done even in the presence of concurrency or parallellism.

lit.: – Jan Tijmen Udding, *Classification and Composition of Delay-Insensitive Circuits*, Eindhoven: University of Technology, September 1984, (Department of Mathematics and Computing Science; doctoral dissertation).

– This thesis.

2 The technique "transformation into computation interference hazard" can be used to transform some liveness properties into safety properties.

lit.: – This thesis.

3 We consider factorization. Factorization is the decomposition problem, in which the specification and a part of the desired solution are given and the remainder has to be calculated. Factorization is equal to composition if and only if all correctness concerns are symmetric w.r.t. the specification and all parts of the solution.

lit.: – This thesis.

4 It is possible to give a formal definition of "the observation of delay-insensitive communication".

5 The so-called "isochronic forks" have been a severe impediment to the development of design methods for circuits with asynchronous communication.

6 Some liveness properties are expressable in finite trace theory. "Absence of ambiguous quiescence hazard" is such a liveness property.

lit.: – Huub M.J.L. Schols, *Notes on Delay-insensitive Communication*, Eindhoven: University of Technology, March 1988, (Department of Mathematics and Computing Science; Computing Science Notes 88/06). [In this paper "ambiguous quiescence hazard" is referred to as "unspecified termination hazard"].

7 For every natural number N, $N \geq 2$, there exist alphabet A and minimal deterministic state graph S, that contains exactly N states, such that projecting S onto A yields a minimal deterministic state graph that contains $(3 * 2^{(N-2)} - 1)$ states. ($(3 * 2^{(N-2)} - 1)$ is the upper limit).

lit.: – Huub M.J.L. Schols, *The Maximum Number of States after Projection*, Eindhoven: University of Technology, April 1987, (Department of Mathematics and Computing Science; Computing Science Notes 87/08).

8 We consider a stack consisting of F fast cells at the top and S slow cells at the bottom. The speed ratio between the fast and the slow cells is r. A sufficient condition for using such a stack at full speed is: $S < (F / (2 * (r - 1)))$.

9 The technique "restoring an invariant", which is often used in sequential programming, suggests a wrong connotation of the notion "invariant", viz. an invariant may not hold; this connotation is an impediment to the construction of invariants for parallel programs.

10 A complete presentation of research does not only contain the final results, but also the discarded results and the reason for discarding them.

11 Engineering is a creative profession. Computer scientists should not try to "automate" engineers. They should rather provide the engineers with tools that enable the engineers to use their creativity.