Classification and composition of delay-insensitive circuits

CLASSIFICATION AND COMPOSITION OF DELAY-INSENSITIVE CIRCUITS

J. T. UDDING

CLASSIFICATION AND COMPOSITION OF DELAY-INSENSITIVE CIRCUITS

PROEFSCHRIFT

TER VERKRIJGING VAN DE GRAAD VAN DOCTOR IN DE
TECHNISCHE WETENSCHAPPEN AAN DE TECHNISCHE
HOGESCHOOL EINDHOVEN, OP GEZAG VAN DE RECTOR
MAGNIFICUS, PROF.DR. S.T.M. ACKERMANS, VOOR
EEN COMMISSIE AANGEWEZEN DOOR HET COLLEGE
VAN DEKANEN IN HET OPENBAAR TE VERDEDIGEN OP
DINSDAG 25 SEPTEMBER 1984 TE 16.00 UUR

DOOR

JAN TIJMEN UDDING

GEBOREN TE DEN HELDER

Dit proefschrift is goedgekeurd
door de promotoren

Prof.dr. M. Rem

en

Prof.dr. E.W. Dijkstra

Druk: Dissertatie Drukkerij Wibro, Helmond, Telefoon 04920-23981.

you only grow by coming to the end of
something and by beginning something new

from 'The World according to Garp'
by John Irving

Contents

0

Introduction

VLSI technology appears to be a powerful medium to realize highly concurrent computations. The fact that we can now fabricate systems that are more complex and more parallel makes high demands, however, upon our ability to design reliable systems. Our main concern in this monograph is to address the problem of specifying components in such a way that, when a number of them is composed using a VLSI medium, the specification of the composite can be deduced from knowledge of the specifications of the components and of the way in which they are interconnected. We confine our attention to temporal and sequential aspects of components and do not, for example, discuss their layouts.

For the specification and composition of components we use a discrete and metric-free formalism, which can be used for the design of concurrent algorithms as well. Therefore, the separation of the design of concurrent algorithms from their implementation as chips, which we have actually introduced in the preceding paragraph, does not seem to move the two too far apart. In fact, we believe that this formalism constitutes a good approach to a mechanical translation of algorithms into chips.

+ + +

A typical VLSI circuit consists of a large number of active electronic elements. It distinguishes itself from LSI circuits by a significantly larger amount of transistors. Unfortunately, existing layouts for circuits cannot simply be mapped onto a smaller area as technology improves. The behaviour of a circuit may change when it is scaled down, since assumptions made for LSI are no longer valid for VLSI. The reason is that parameters determining a circuit's behaviour do not scale in the same way, when the size of that circuit is scaled down.

As has been argued in [9], scaling down a circuit's size by dividing all dimensions by a factor α results in a transit time of the transistors that is α times shorter. The propagation time for an electrical signal between two points on a wire, however, is the same as the propagation time for an electrical signal between the two corresponding points in the scaled circuit. In VLSI circuits the relationship between delay and transit time becomes such that delays of signals in connecting wires might not be neglected anymore.

From the above we conclude that, if we want circuit design to be independent of the circuit's size, we have to employ a method that relies neither upon the speed with which a component or its environment responds nor upon the propagation delay of a signal along a connecting wire. The resulting kind of components we call delay-insensitive. Another advantage of delay-insensitive components is that we have a greater layout freedom, since the lengths of connecting wires are no longer relevant to correctness of operation.

Apart from the reasons mentioned above, there is yet another motive for the design of delay-insensitive circuits. In a lot of concurrent computations a so-called arbitration device is used. Basically, such a device grants one out of several requests. Real-time interrupts are a typical example of the use of such a device. In its simplest form it can be viewed as a bistable device. Consequently, under some continuity assumptions [4], it has a metastable state. The closer its initial state is to the metastable state the longer it takes before it settles down in one of its stable states. Starting from the metastable state it even may never end up in a stable state.

In clocked systems, where all computational units are assumed to complete each of their computations within a fixed and bounded amount of time, this so-called glitch phenomenon may lead to malfunctioning. This problem was first signalled in the late sixties [0,8]. The only way to guarantee fully correct communications with an arbitration device is to make the communicating parts delay-insensitive. This is not the way, however, in which this problem is solved in present-day computers, where the probability of correct communications is made sufficiently large, by allowing, for example, on the average one failure of this kind a year. This is achieved by reducing the clock rate and, hence, the computation speed. From an industrial point of view this may be quite satisfactory. From a theoretical point of view it certainly is not.

+ + +

In this monograph the foundation of a theory on delay-insensitive circuits is laid. The notion of delay-insensitivity is formally defined and a classification of delay-insensitive components is given in an axiomatic way. Moreover, a composition operator for these components is introduced and its correctness is discussed. Crucial to this discussion is that we do not want to assume anything about absolute or relative delays in wires that connect these components, except that delays are non-negative. This leads to two conditions that should be complied with upon composition.

First, in order to prevent a voltage level transition from interfering with another one propagating along the same wire at most one transition is allowed to be on its way along a wire, since successive voltage level transitions may propagate at different speeds. At best, this kind of interference leads to absorption of transitions, which can be viewed as an infinite delay. At worst, however, it causes the introduction of new transitions, which may lead to malfunctioning. Therefore, absence of transmission interference is to be guaranteed upon composition.

Second, we have to guarantee absence of computation interference. Computation interference is the arrival of a voltage level transition at a circuit before that circuit is ready - according to its specification - to receive it. In other words, an input signal should not interfere with the computation that goes on before the circuit is ready for that signal's reception. Due to unknown wire delays, this amounts to not sending a signal before the receiver is ready for it.

+ + +

How to get delay-insensitive circuits in the first place is not a topic addressed here. One can follow the method proposed by Seitz [11] and divide a chip into so-called isochronic regions. These regions are so small that, within a region, the wire delays are negligibly small. They are then interconnected by wires with delays about which no assumptions are made. The smaller the regions are chosen the less sensitive such circuits will be to scaling. Another method is the one proposed by Fang and Molnar [3]. They model a circuit as a Huffman asynchronous sequential circuit with certain of its inputs consisting of the fed-back values of some of its outputs. Then it can be shown that the circuit thus obtained is delay-insensitive in its communications with the environment, provided that both the combinational circuit and the internal delays meet certain conditions.

A communication protocol that is often used for databuses [13] allows a number of voltage level transitions to occur on a wire before the final level on that wire represents a signal and can be inspected. The presence of such a final level on a wire is then signalled by a so-called data valid signal, for which a second wire is used. For this kind of protocol, however, we need to know something about the relative wire delays, for which, for example, so-called bundling constraints can be used. As pointed out above, we do not want to assume anything about absolute or relative wire delays and, hence, we do not investigate this kind of protocol. The approach that is advocated here is first to understand the composition of fully delay-insensitive circuits and next to decide whether and how to incorporate items like bundling constraints. Consequently, we assume transitions from one voltage level to another to be monotonic.

+ + +

In the first chapter we summarize trace theory and discuss a composition operator, blending, in particular. A more comprehensive discussion can be found in [12]. Trace theory is a discrete and metric-free formalism, in which we can adequately define notions such as delay-insensitivity and absence of computation and transmission interference. In the subsequent chapter we define and classify delay-insensitive components. This classification is illustrated by a number of examples. The third chapter is devoted to partitioning the wires of these components into independent groups via which composition is possible. In addition, we state a number of conditions that must be satisfied if this composition is to be allowed. In the subsequent chapter it is argued that, under these conditions, there is no computation and transmission interference. Moreover, it turns out that we can specify the composite by means of the blend of the specifications of the composing parts. The fifth chapter shows which of the classes introduced in Chapter 2 are closed under this composition operator. Finally, in Chapter 6, some clues are given to relax the composition conditions in order to incorporate other, more general, kinds of compositions for delay-insensitive circuits than the ones discussed here.

+ + +

A slightly unconventional notation for variable-binding constructs is used. It will be explained here informally. Universal quantification is denoted by

$$(\forall l \; : D \; : E \;)$$

where $\forall$ is the quantifier, l is a list of bound variables, D is a predicate, and E is the quantified expression. Both D and E will, in general, contain variables from l. D delineates the domain of the bound variables. Expression E is defined for variable values that satisfy D. Existential quantification is denoted in a similar way with quantifier $\exists$. In the case of set formation we write

$$\{ \; l : D \; : E \; \}$$

to denote the set of all values of E obtained by substituting for all variables in l values that satisfy D. The domain D is omitted when obvious from the context.

For expressions E and G, an expression of the form $E \Rightarrow G$ will often be proved in a number of steps by the introduction of intermediate expressions. For instance, we can prove $E \Rightarrow G$ by proving $E = F$ and $F \Rightarrow G$ for some expression F. In order not to be forced to write down expressions like F twice, expressions that often require a lot of paper, we record proofs like this as follows.

E

$=$ { hint why $E = F$ }

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

We shall frequently use the hint calculus, viz. when appealing to everyday mathematics, i.e. predicate calculus, arithmetics, and, above all, common sense.

These notions have been adapted from [2].

1

Trace theory

In order to define and classify delay-insensitive circuits we need a formalism for their specification. For that purpose we use trace theory. In the present chapter we give an overview of trace theory as far as we need it for this monograph. A more thorough discussion can be found in [12].

1.0. Traces and trace structures

An alphabet is a finite set of symbols. Symbols are denoted by identifiers. For each alphabet A, A^* denotes the set of all finite-length sequences of elements of A, including the empty sequence, which is denoted by ϵ. Finite-length sequences of symbols are called traces. A trace structure T is a pair $<U,A>$, in which A is an alphabet and U a set of traces satisfying $U \subseteq A^*$. U is called the trace set of T and A is called the alphabet of T. The elements of U are called traces of T and the elements of A are called symbols of T.

We postulate operators **t**, **a**, **i**, and **o** on trace structures. For trace structure T, $\mathbf{t}\,T$ and $\mathbf{a}\,T$ are the trace set of T and the alphabet of T respectively. $\mathbf{i}\,T$ and $\mathbf{o}\,T$ are disjoint subsets of $\mathbf{a}\,T$. $\mathbf{i}\,T$ is called the input alphabet of T and $\mathbf{o}\,T$ the output alphabet. Notice that $\mathbf{i}\,T \cup \mathbf{o}\,T$ need not be equal to $\mathbf{a}\,T$.

An informal mechanistic appreciation of a trace structure is the following. A trace structure is viewed as the specification of a mechanism communicating with its environment. Symbols of the trace structure's alphabet are the various kinds of communication actions possible between mechanism and environment. The input symbols of the trace structure are inputs with respect to the mechanism and outputs with respect to the environment. The output symbols of the trace structure are outputs with respect to the mechanism and inputs with respect to the environment. A trace structure's trace set is the set of all possible sequences of communication actions that can take place between the mechanism and its environment.

With a mechanism in operation we associate a so-called trace thus far generated. This is a trace of the trace structure of that mechanism. Initially the trace thus far generated is ϵ, which apparently belongs to the trace structure. Each act of communication corresponds to extending the trace thus far generated with the symbol associated with that act of communication.

This appreciation pertains to a mechanism more abstract than an electrical circuit. It enables us to explore in the next two chapters properties that may be associated with delay-insensitivity. In Chapter 4, finally, we are able to give a mechanistic appreciation of trace structures that is tailored to electrical circuits.

Example 1.0

A Wire is allowed to convey at most one voltage level transition. We assume that there are two voltage levels, viz. low and high. Hence, we can view a wire as a mechanism that is able to accept either a voltage level transition from low to high, whereafter it produces the same transition at its output, or to accept a voltage level transition from high to low, whereafter it produces that transition at its output again. Since the two kinds of transitions alternate, we do not make a distinction in our formalism between a high-going and a low-going transition. Consequently, the specification of such a wire is a trace structure with input alphabet $\{ a \}$, output alphabet $\{ b \}$, and trace set the set of all finite-length alternations of a and b that do not start with b.

(End of Example)

Note : Unless stated otherwise, small and capital letters near the end of the Latin alphabet are used to denote traces and trace structures respectively. Small and capital letters near the beginning of the Latin alphabet denote symbols and alphabets respectively.

(End of Note)

The projection of trace t on alphabet A, denoted by $t \lceil A$, is defined as follows

$$\begin{array}{l} \text{if } t = \epsilon \text{ then } t \lceil A = \epsilon \\ \text{if } t = ua \wedge a \in A \text{ then } t \lceil A = (u \lceil A)a \\ \text{if } t = ua \wedge a \notin A \text{ then } t \lceil A = (u \lceil A) \end{array}$$

(concatenation is denoted by juxtaposition.)

The projection of a trace set T on alphabet A, denoted by $T\lceil A$, is the trace set $\{t : t \in T : t\lceil A\}$ and the projection of trace structure T on A, denoted by $T\lceil A$, is the trace structure $<(\mathbf{t}T)\lceil A, \mathbf{a}T \cap A>$. The input alphabet $\mathbf{i}(T\lceil A)$ and the output alphabet $\mathbf{o}(T\lceil A)$ of $T\lceil A$ are defined as $\mathbf{i}T \cap A$ and $\mathbf{o}T \cap A$ respectively.

Property 1.0 : Projection distributes over concatenation, i.e. for traces t and u, and for alphabet A $\quad (tu)\lceil A = (t\lceil A)(u\lceil A)$.

Property 1.1 : For trace t and alphabets A and B $\quad t\lceil A\lceil B = t\lceil(A \cap B)$.

In order to save on parentheses we give unary operators the highest binding power, and write $\mathbf{t}T\lceil A$ instead of $(\mathbf{t}T)\lceil A$. Moreover, concatenation has a higher binding power than projection. As a consequence, we write $tu\lceil A$ instead of $(tu)\lceil A$.
For trace t the length of t is denoted by $\mathbf{l}t$. For trace t and symbol a $\#_a t$ denotes the number of occurrences of a in t. We call trace s a prefix of trace t if $(\exists u :: su = t)$. For trace set T, the trace set that contains all prefixes of traces of T is called the prefix-closure of T, and is denoted by $\mathbf{pref}T$. A trace set T is called prefix-closed if $T = \mathbf{pref}T$.

Property 1.2 : For prefix-closed trace set T and alphabet A $\quad T\lceil A$ is prefix-closed.

There are two composition operators that we shall frequently use. The first one is weaving. It can, for the time being, be appreciated as the composition of two mechanisms where each communication in the intersection of the two alphabets is the same for both mechanisms. This leads to the following definition. The weave of two trace structures S and T, denoted by $S \mathbf{w} T$, is the trace structure

$$< \{x : x \in (\mathbf{a}S \cup \mathbf{a}T)^* \wedge x\lceil \mathbf{a}S \in \mathbf{t}S \wedge x\lceil \mathbf{a}T \in \mathbf{t}T : x\}, \mathbf{a}S \cup \mathbf{a}T >$$

Input and output alphabet of $S \mathbf{w} T$ are defined as $(\mathbf{i}S \cup \mathbf{i}T)\setminus(\mathbf{a}S \cap \mathbf{a}T)$ and $(\mathbf{o}S \cup \mathbf{o}T)\setminus(\mathbf{a}S \cap \mathbf{a}T)$ respectively. Apparently, the type of non-common symbols does not change and common symbols loose their types.

Example 1.1
$<\{ab, abe, de\}, \{a, b, d, e\}> \mathbf{w} <\{bc, bec, fe\}, \{b, c, e, f\}> \;=$
$<\{abc, abec, dfe, fde\}, \{a, b, c, d, e, f\}>$
(End of Example)

Property 1.3 : For trace structures S and T, for traces s and t, and for symbols $a \in \mathbf{a}S \setminus \mathbf{a}T$ and $b \in \mathbf{a}T \setminus \mathbf{a}S$

$$sabt \in \mathbf{t}(S \mathbf{w} T) \;=\; sbat \in \mathbf{t}(S \mathbf{w} T)$$

Proof :

$sabt \in \mathbf{t}(S \mathbf{w} T)$
$=$ { definition of weaving }
$sabt \in (\mathbf{a}S \cup \mathbf{a}T)^* \;\wedge\; sabt \lceil \mathbf{a}S \in \mathbf{t}S \;\wedge\; sabt \lceil \mathbf{a}T \in \mathbf{t}T$
$=$ { Property 1.0, the distribution of projection over concatenation, using $a \lceil \mathbf{a}T = \epsilon$ and $b \lceil \mathbf{a}S = \epsilon$ }
$sabt \in (\mathbf{a}S \cup \mathbf{a}T)^* \;\wedge\; sat \lceil \mathbf{a}S \in \mathbf{t}S \;\wedge\; sbt \lceil \mathbf{a}T \in \mathbf{t}T$
$=$ { Distribution of projection over concatenation, using $a \lceil \mathbf{a}T = \epsilon$ and $b \lceil \mathbf{a}S = \epsilon$ }
$sbat \in (\mathbf{a}S \cup \mathbf{a}T)^* \;\wedge\; sbat \lceil \mathbf{a}S \in \mathbf{t}S \;\wedge\; sbat \lceil \mathbf{a}T \in \mathbf{t}T$
$=$ { definition of weaving }
$sbat \in \mathbf{t}(S \mathbf{w} T)$

(End of Proof)

Property 1.4 : Weaving is symmetric.

Property 1.5 : The trace set of the weave of two trace structures with prefix-closed trace sets is prefix-closed.

The second operator that we discuss is blending. A weave still reflects the composite's internal structure. By projection on the alphabets of the composing trace structures, the individual traces from which the traces of the composite are formed can be retrieved. After projection on the symmetric difference of the alphabets of the composing trace structures the internal communications are hidden. This blend of two trace structures S and T, denoted by $S \mathbf{b} T$, is the trace structure

$$(S \mathbf{w} T) \lceil (\mathbf{a}S \div \mathbf{a}T)$$

where $\div$ denotes symmetric set difference. Input and output alphabet of $S \mathbf{b} T$ are defined as $\mathbf{i}(S \mathbf{w} T)$ and $\mathbf{o}(S \mathbf{w} T)$ respectively.

Property 1.6 : Blending is symmetric.

Example 1.2 (cf. Example 1.1)
$< \{ ab, abe, de \}, \{ a, b, d, e \} > \mathbf{b} < \{ bc, bec, fe \}, \{ b, c, e, f \} > \ =$
$< \{ ac, df, fd \}, \{ a, c, d, f \} >$
(End of Example)

Property 1.7 : The trace set of the blend of two trace structures with prefix-closed trace sets is prefix-closed.

Property 1.8 : For trace structures S and T and for trace s

$$s \in \mathbf{t}(S \mathbf{b} T) \Rightarrow s \lceil (\mathbf{a}S \setminus \mathbf{a}T) \in \mathbf{t}S \lceil (\mathbf{a}S \setminus \mathbf{a}T)$$

Proof :

$s \in \mathbf{t}(S \mathbf{b} T)$
$=$ { definition of blending }
$(\exists s_0 :: s_0 \in \mathbf{t}(S \mathbf{w} T) \wedge s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s)$
$\Rightarrow$ { definition of weaving }
$(\exists s_0 :: s_0 \lceil \mathbf{a}S \in \mathbf{t}S \ \wedge s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s)$
$\Rightarrow$ { projection on $\mathbf{a}S \setminus \mathbf{a}T$ and Property 1.1, using
$\mathbf{a}S \cap (\mathbf{a}S \setminus \mathbf{a}T) = (\mathbf{a}S \div \mathbf{a}T) \cap (\mathbf{a}S \setminus \mathbf{a}T)$ }
$(\exists s_0 :: s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) \lceil (\mathbf{a}S \setminus \mathbf{a}T) \in \mathbf{t}S \lceil (\mathbf{a}S \setminus \mathbf{a}T) \wedge s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s)$
$\Rightarrow$ {calculus }
$s \lceil (\mathbf{a}S \setminus \mathbf{a}T) \in \mathbf{t}S \lceil (\mathbf{a}S \setminus \mathbf{a}T)$

(End of Proof)

1.1. A program notation

In this section we discuss a way to represent trace structures. Since trace sets are often infinite, a representation by enumeration of its elements becomes rather cumbersome. We use so-called commands with which we associate trace structures.
With command S trace structure $\mathbf{TR}\,S$ is associated in the following way.

- A symbol is a command. For symbol a $\quad \mathbf{TR}\,a \;=\; <\{a\},\{a\}>$.
- If S and T are commands then $(S \mid T)$ is a command.
  $\mathbf{TR}(S \mid T) \;=\; <\mathbf{t}(\mathbf{TR}\,S) \cup \mathbf{t}(\mathbf{TR}\,T)\,,\,\mathbf{a}(\mathbf{TR}\,S) \cup \mathbf{a}(\mathbf{TR}\,T)>$.
- If S and T are commands then $(S\,;T)$ is a command. $\mathbf{TR}(S\,;T) =$
  $<\{x,y : x \in \mathbf{t}(\mathbf{TR}\,S) \wedge y \in \mathbf{t}(\mathbf{TR}\,T) : xy\}\,,\,\mathbf{a}(\mathbf{TR}\,S) \cup \mathbf{a}(\mathbf{TR}\,T)>$.
- If S and T are commands then $(S\,,T)$ is a command.
  $\mathbf{TR}(S\,,T) \;=\; (\mathbf{TR}\,S)\,\mathbf{w}\,(\mathbf{TR}\,T)$.
- If S is a command then S^* is a command.
  $\mathbf{TR}(S^*) \;=\; <(\mathbf{t}(\mathbf{TR}\,S))^*\,,\,\mathbf{a}(\mathbf{TR}\,S)>$

Furthermore, there are a few priority rules. The star has the highest priority, followed by the comma, the semicolon, and the bar. The trace sets thus obtained are not prefix-closed. Since we are interested in prefix-closed trace sets only, as will turn out in the next chapter, we associate with a command S the trace structure $<\mathbf{pref}(\mathbf{t}(\mathbf{TR}\,S)), \mathbf{a}(\mathbf{TR}\,S)>$, when the command is used for the specification of a mechanism.

Example 1.3

The specification of a Wire, as exemplified in Example 1.0 would be : input alphabet $\{a\}$, output alphabet $\{b\}$, and command $(a\,;b)^*$.
(End of Example)

Example 1.4

A Muller-C element, or C-element for short [6], is an element with two inputs and one output. It is supposed to synchronize the inputs, i.e. after having received an input change on both input wires, it produces a change on the output wire. Its specification is a trace structure with input alphabet $\{a,b\}$, output alphabet $\{c\}$, and command $(a\,,b\,;c)^*$.
(End of Example)

2

Classification of delay-insensitive trace structures

With the trace theory as introduced in the preceding chapter we are now able to define delay-insensitive trace structures formally. We reserve the term component for a mechanism that is an abstraction of an electrical circuit. A trace structure is the specification of the communications between a component and its environment. Inputs of the trace structure are inputs with respect to the component and outputs with respect to the environment. Outputs of the trace structure are outputs with respect to the component and inputs with respect to the environment.

The key to the definition of delay-insensitive trace structures is the component and its environment being insensitive to the speeds with which they operate and to propagation delays in connecting wires. This is informally captured by viewing a component as being wrapped in some kind of foam box representing a flexible and possibly time-varying boundary. The communication actions between component and environment are specified at this boundary. The flexibility of this boundary imposes certain restrictions that the specification of a delay-insensitive circuit has to satisfy. As will turn out in the sequel, these requirements basically amount to the absence of ordering between certain symbols : the presence of certain traces in a trace structure's trace set implies the presence of other traces in that trace set. It is not a priori obvious that the requirements deduced in this chapter on account of this foam rubber wrapper principle are sufficient to guarantee proper communications. This will only turn out in Chapter 4.

The first restriction to be imposed upon a trace structure is that its alphabet be partitioned into an input and an output alphabet. We do not, at this level of abstraction at least, consider a communication means other than input or output, nor do we consider ports that are input at one time and output at another time. This means that we have for trace structure T the rule

$\mathbf{R}_0$) $\mathbf{i}T \cup \mathbf{o}T = \mathbf{a}T$

Notice that $\mathbf{i}T \cap \mathbf{o}T = \emptyset$ according to the definition of a trace structure.

Second, we impose the restriction that a trace set be prefix-closed and non-empty. This rule is dictated by the fact that a system that can produce trace ta is assumed to do so by first producing t and then a. The symbols in a trace structure's alphabet are viewed as atomic actions. Moreover, a system must be able to produce ϵ initially. This gives for trace structure T the rule

$\mathbf{R}_1$) $\mathbf{t}T$ is prefix-closed and non-empty

The basic idea of this monograph is that we do not make any assumptions on absolute or relative wire delays. As we pointed out in the introduction, this leads to the assumption of a transition being monotonic in order to enable a component to recognize the signal that this transition represents. This means that we have to guarantee transitions against interference and, therefore, have to limit the number of transitions on a wire to at most one. In terms of trace structures, where signals via the same wire are represented by the same symbol, this amounts to the restriction that adjacent symbols be different. This gives for trace structure T the following necessary condition.

$\mathbf{R}_2$) for trace s and symbol $a \in \mathbf{a}T$ $\quad saa \notin \mathbf{t}T$

Signals are sent in either of two directions, viz. from a component to its environment or the other way round. Due to unknown wire delays, two signals being sent the one after the other in the same direction via different wires need not be received in the order in which they are sent. In other words, we cannot assume our communications to be order preserving. Consequently, a specification of a delay-insensitive component does not depend on the order in which this kind of concurrent signals is sent or received. Therefore, a trace structure containing a trace with two adjacent symbols of the same type (input or output) also contains the trace with these two symbols swapped. In fact, we conceive adjacent symbols of the same type as not being ordered at all. (Their occurrence as adjacent symbols in a trace is just a shortcoming of our writing in a linear way.) For trace structure T, this is expressed by the following restriction

$\mathbf{R}_3$) for traces s and t, and for symbols $a \in \mathbf{a}T$ and $b \in \mathbf{a}T$ of the same type
$sabt \in \mathbf{t}T = sbat \in \mathbf{t}T$

Due to the foam rubber wrapper principle, signals in opposite directions are subject to restrictions as well. As opposed to signals of the same type, they may have a causal relationship and, hence, have an order. If, however, in some phase of the computation they are not ordered, meaning that for some trace s and symbols a and b both $sa \in \mathbf{t}T$ and $sb \in \mathbf{t}T$, then the traces that sab and sba can be extended with, according to the component's trace set, should not differ too much. Obviously, we do justice to the foam rubber wrapper principle if the order of this kind of concurrent symbols is of no importance at all. This results for trace structure T in the rule

$\mathbf{R}_4'$) for traces s and t, and for symbols $a \in \mathbf{a}T$ and $b \in \mathbf{a}T$ of different types
$sa \in \mathbf{t}T \wedge sbat \in \mathbf{t}T \Rightarrow sabt \in \mathbf{t}T$

Finally, we have to take into account that a signal, once sent, cannot be cancelled. However long it takes, eventually it will reach its destination. Consequently, a component ready to receive a certain signal from its environment, which means that the trace thus far generated extended with that symbol belongs to the trace set, must not change its readiness when sending a signal to its environment. In other words, in the absence of an oracle informing either side on signals that, though possible, will not be sent, we cannot allow in a specification that a symbol disables a symbol of another type. Symbol a disables symbol b in trace structure T if there is a trace s with

$$sa \in \mathbf{t}T \wedge sb \in \mathbf{t}T \wedge sab \notin \mathbf{t}T$$

There is nothing wrong, however, with symbols that disable symbols of the same type. If these symbols are input symbols then the environment has to make a decision which output symbol(s) to send. If, on the contrary, the symbols are output symbols then the component has to make that decision. Since a correct use of arbitration devices is one of the important incentives to the study of delay-insensitive circuits, the various types of decisions are a key to the classification. Three classes, each of them described by one of the following non-disabling rules, can be distinguished now. For trace structure T we have

$\mathbf{R}_5'$) for trace s and distinct symbols $a \in \mathbf{a}T$ and $b \in \mathbf{a}T$
$sa \in \mathbf{t}T \wedge sb \in \mathbf{t}T \Rightarrow sab \in \mathbf{t}T$

$\mathbf{R}_5''$) for trace s and distinct symbols $a \in \mathbf{a}T$ and $b \in \mathbf{a}T$, not both input symbols, $sa \in \mathbf{t}T \wedge sb \in \mathbf{t}T \Rightarrow sab \in \mathbf{t}T$

$\mathbf{R}_5'''$) for trace s and symbols $a \in \mathbf{a}T$ and $b \in \mathbf{a}T$ of different types
$sa \in \mathbf{t}T \wedge sb \in \mathbf{t}T \Rightarrow sab \in \mathbf{t}T$

All delay-insensitive trace structures satisfy $\mathbf{R}_0$ through $\mathbf{R}_3$. The class satisfying $\mathbf{R}_4'$ and $\mathbf{R}_5'$ as well is called the synchronization class. It is also denoted by $\mathbf{C}_1$. A specification in this class allows for synchronization only. Due to the absence of decisions, no data transmission is possible. The class allowing for input symbols to be disabled, satisfying therefore $\mathbf{R}_4'$ and $\mathbf{R}_5''$, is called the data communication class. It is also denoted by $\mathbf{C}_2$. Here the data is encoded by means of the possible decisions. Finally, we have $\mathbf{C}_3$, or the arbitration class, which allows a component to choose between output symbols. Specifications in this class satisfy, in addition to $\mathbf{R}_0$ through $\mathbf{R}_3$, $\mathbf{R}_4'$ and $\mathbf{R}_5'''$. Obviously, $\mathbf{C}_1 \subset \mathbf{C}_2 \subset \mathbf{C}_3$.

We could have distinguished the class in which decisions are made in the component and not in the environment, which is $\mathbf{C}_2$ with in its $\mathbf{R}_5''$ the restriction 'not both inputs' replaced by 'not both outputs'. We have not done so, however, since none of the classes thus obtained turns out to be closed under the composition operator proposed in the next chapter, a circumstance making none of these classes very interesting. $\mathbf{C}_3$ has, arbitrarily, been chosen to demonstrate this phenomenon.

The reason that $\mathbf{C}_3$ is not closed under composition is that $\mathbf{R}_4'$ is too restrictive in the presence of decisions in the component, as is shown in Chapter 5. We concluded the analysis for $\mathbf{R}_4'$ by observing that the foam rubber wrapper principle would certainly be done justice if the order of concurrent symbols of different types was of no importance. This situation, however, needs a more careful analysis.

The specification of a component must not depend on the place of the boundary of the foam rubber wrapper. Consider two wrappers, the one contained in the other one. If, at the outside boundary the order between two concurrent input and output signals is input-before-output, then nothing can be said about their order at the inside boundary. If, on the other hand, the order between such signals is output-before-input at the outside boundary, then the same order between these symbols is implied at the inside boundary.

The first situation, i.e. input-before-output at the outside boundary, gives rise to a restriction to be imposed upon a component's trace set. Assume that we have traces s and t, input symbol a, output symbol b, and traces $sabt$ and $sbat$ in the component's trace set. Trace $sabt$ is the trace associated with the outside boundary and trace $sbat$ is the one that is associated with the inside boundary. Now if $sabt$ can be extended -according to the component's trace set- with an input symbol c, which means a signal from the outside boundary towards the inside boundary, then a necessary condition for absence of computation interference at the inside boundary is the presence of trace $sbatc$ in the component's trace set.

A similar observation applies to an output-before-input order of concurrent symbols at the inside boundary and an input-before-output order at the outside boundary. In this case an output signal possible at the inside boundary should be possible at the outside boundary as well. This results for trace structure T in the following rule, which is less restrictive than $\mathbf{R}_4'$.

$\mathbf{R}_4''$) for traces s and t, and for symbols $a \in \mathbf{a}T$, $b \in \mathbf{a}T$, and $c \in \mathbf{a}T$ with b of another type than a and c $\quad sabtc \in \mathbf{t}T \wedge sbat \in \mathbf{t}T \Rightarrow sbatc \in \mathbf{t}T$

$\mathbf{R}_0$ through $\mathbf{R}_3$ together with $\mathbf{R}_4''$ and any of the three $\mathbf{R}_5$'s constitute a class of delay-insensitive trace structures. We give a name to the largest class only, which is the one with $\mathbf{R}_4''$ and $\mathbf{R}_5'''$. We call it the class of delay-insensitive trace structures and denote it by $\mathbf{C}_4$. Obviously, $\mathbf{C}_3 \subset \mathbf{C}_4$. We do not attach names to the other classes, since these classes neither provide more insight nor have surprising properties.

Before exploring $\mathbf{R}_4''$, we illustrate this classification by a number of examples. In these examples we sometimes represent a trace structure by a state graph instead of by a command. A state graph is a directed graph with one special node, the start node, and arcs labelled with symbols of the trace structure's alphabet. Each path from the start node corresponds to a trace, viz. the one that is brought about by the labels of the consecutive arcs in that path. A state graph is said to represent a trace structure if it has the same trace set as that trace structure. Rules $\mathbf{R}_3$, $\mathbf{R}_4'$, and $\mathbf{R}_5$ are usually more easily checked in a state graph than in a command. Rule $\mathbf{R}_4''$ is hard to check in either representation. In the figures the start nodes are drawn fat. Choosing another node as start node means another initialization of the component. Components that only differ from one another by different start nodes are given the same name. For clearness' sake we attach a question mark to arcs labelled with an input symbol and an exclamation mark to arcs labelled with an output symbol.

Example 2.0

The Wire and the C-element of Examples 1.3 and 1.4 are $\mathbf{C}_1$'s. Interchanging the roles of the input and the output alphabet yields $\mathbf{C}_1$'s again. The wire remains a wire, now starting with an output however. The C-element becomes a Fork, viz. a trace structure with input alphabet $\{c\}$, output alphabet $\{a, b\}$, and command $(a, b\,; c)^*$. By another initialization we also have the command $(c\,; a, b)^*$ for a Fork.

(End of Example)

Example 2.1

Another very common element is the so-called Merge. It is an element with input alphabet $\{a, b\}$, output alphabet $\{c\}$, and command $((a \mid b); c)^*$. This component is a $\mathbf{C}_2$, since inputs a and b disable one another. Interchanging the roles of input and output alphabet yields a $\mathbf{C}_3$. This is the simplest form of an arbiter.

(End of Example)

Example 2.2

A C-element with two outputs instead of one is another example of a $\mathbf{C}_1$. It has input alphabet $\{a, b\}$ and output alphabet $\{c, d\}$. There are two essentially different trace structures that synchronize the input signals. The first one is the C-element with its output symbol replaced by two output symbols in any order. This yields command $(a, b\,;c, d)^*$. In this trace structure we can distinguish an input and an output phase. Another command allows the two phases to overlap a little bit, but still synchronizes the inputs. This is expressed in the command $a, b\,;((c\,;a),(d\,;b))^*$.

(End of Example)

Example 2.3

Consider a C-element with input alphabet $\{a, r\}$, output alphabet $\{p\}$, and command $(a, r\,;p)^*$ and consider a Wire with input alphabet $\{q\}$, output alphabet $\{b\}$, and command $(q\,;b)^*$. The Wire can be used to acknowledge the reception of symbol p by the environment before a next input a is allowed to occur. The resulting component has input alphabet $\{a, q, r\}$, output alphabet $\{b, p\}$, and command $a\,;(p\,;(q\,;b\,;a), r)^*$. We have chosen this initialization, since the component will be used in this form in Chapter 5. It is a $\mathbf{C}_1$.

(End of Example)

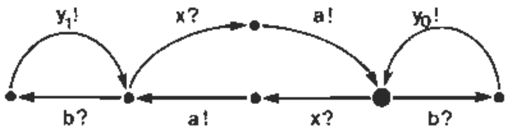
Example 2.4

Another component that will be used in Chapter 5 is a component that can be thought of as consisting of three wires : two wires to convey a bit of information and one wire for the acknowledgement of its arrival. A bit is encoded as sending a signal on one of the two wires that are used for the data transmission. Its input alphabet is $\{x_0, x_1, b\}$, its output alphabet is $\{y_0, y_1, a\}$, and its command is $(x_0; y_0\,; b\,; a \mid x_1; y_1\,; b\,; a)^*$. Because of the choice to be made between the inputs x_0 and x_1 this component is a $\mathbf{C}_2$.

(End of Example)

Example 2.5

A parity counter is a component that counts the parity of a number of consecutive inputs. The parity can be retrieved on request an unbounded number of times. The symbol whose occurrences we want to count is x. Its reception by the component is acknowledged by symbol a. By means of symbol b we can retrieve the parity of the occurrences of x so far. Symbol y_0 represents an even number and symbol y_1 an odd number of occurrences. The trace structure's input alphabet is $\{x, b\}$, its output alphabet is $\{a, y_0, y_1\}$, and its command is $((b; y_0)^*; x; a; (b; y_1)^*; x; a)^*$. This component is a $\mathbf{C}_2$. There is a choice to be made between inputs x and b. To show that more clearly we draw a state graph of this component.



Any two arcs from the same node have labels of the same type, which implies that $\mathbf{R}_4'$ is trivially satisfied. There are no two consecutive arcs with labels of the same type, which implies that $\mathbf{R}_3$ is satisfied. Any two arcs from the same node have labels of type input that do disable one another. This does not meet requirement $\mathbf{R}_5'$, but this is allowed according to $\mathbf{R}_5''$. Consequently, this is a $\mathbf{C}_2$.

(End of Example)

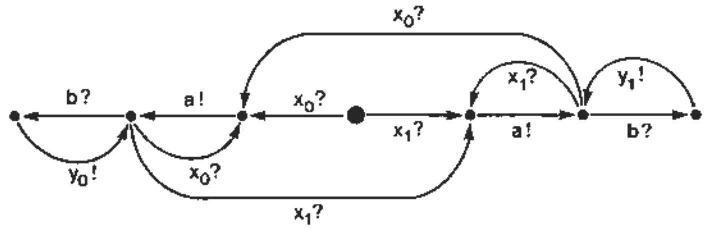
Example 2.6

An And-element with input alphabet $\{a, r\}$ and output alphabet $\{c\}$ is quite often used in the following way. Both inputs go high in some order whereafter the output follows the inputs. Next, both inputs go low again and the output follows the first low-going input transition. This is expressed by the command $(a, r; c; (a; (c, r) \mid r; (a, c))^*$. This trace structure is not delay-insensitive, however. It contains, for instance, the trace *arcrcaa*, which violates $\mathbf{R}_2$. It can be made delay-insensitive by replicating both inputs. Then its input alphabet is $\{a, r\}$, its output alphabet $\{b, c, p\}$ and a possible command $(a; p; r; b, c; a; c, (p; r; b))^*$. Input a is now acknowledged by p and r by b. It is not the most general command for a delay-insensitive And-element but one that suffices for the sequel. The corresponding trace structure is a $\mathbf{C}_1$.

(End of Example)

Example 2.7

A binary variable is a component that can store one bit of information, which may be retrieved afterwards on request an unbounded number of times. The component has input alphabet $\{x_0, x_1, b\}$, output alphabet $\{y_0, y_1, a\}$, and command $(x_0; a; (b; y_0)^* \mid x_1; a; (b; y_1)^*)^*$. Symbol a acknowledges the reception of a bit (either x_0 or x_1), and b is the request for the currently stored value. A state graph looks like



In the start node a choice has to be made between x_0 and x_1 (it has no currently stored value). Moreover, there are two nodes where a choice has to be made between inputs b, x_0, and x_1. This makes it a $\mathbf{C}_2$.

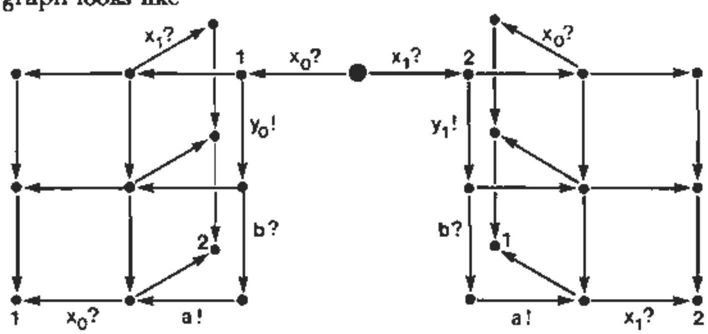
(End of Example)

Example 2.8

A buffer is an element that allows us to store a series of values and to retrieve them in the same order. Usually a buffer has a finite number of places for storage, which bounds the number of values that can be stored simultaneously. In this example we discuss a one-place one-bit buffer. The reception of one bit, either x_0 or x_1, is acknowledged by a. Symbols y_0 and y_1 are used to return the stored value. Symbol b signals the environment's readiness (or request) for the next value. Initially the environment is ready to receive a value. There exist less complicated buffers, more similar to the variable of the preceding example. We have chosen for this buffer and this initialization, since this buffer can easily be composed with another one as will turn out in Chapters 3 and 5. The trace structure of this component has input alphabet $\{x_0, x_1, b\}$, output alphabet $\{y_0, y_1, a\}$, and command

$$x_0 ; (((a ; x_0),(y_0 ; b))^* ; (a ; x_1),(y_0 ; b) ; ((a ; x_1),(y_1 ; b))^* ; (a ; x_0),(y_1 ; b))^* \mid$$
$$x_1 ; (((a ; x_1),(y_1 ; b))^* ; (a ; x_0),(y_1 ; b) ; ((a ; x_0),(y_0 ; b))^* ; (a ; x_1),(y_0 ; b))^*$$

A state graph looks like

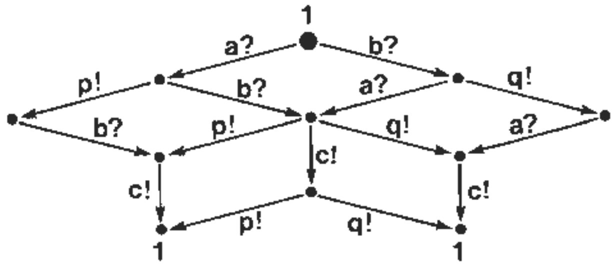


We have not labelled all arcs. Opposite sides of the parallelograms have equal labels. Nodes that have been attached the same number are identical. Here we see the existence of a node with outgoing arcs with labels of different types. It is easy to see that $\mathbf{R}_4'$ is still satisfied, since arcs with such labels make up a parallelogram, which means that their order is of no importance. This component is a $\mathbf{C}_2$, the only decision to be made being the one between inputs x_0 and x_1.

(End of Example)

Example 2.9

An arbiter, in one of its simplest forms, grants one out of two requests. The arbiter that we discuss in this example has a cyclic way of operation, i.e. it needs both requests before being able to deal with the next request. It has input alphabet $\{ a, b \}$ and output alphabet $\{ c, p, q \}$. In every cycle exactly one of the outputs p and q changes. A change in a precedes a change in p and, likewise, a change in q is preceded by a change in b. The output c signals the completion of the cycle after reception of a and b. Consequently, the command is $((a,b\ ;c),((a\ ;p),b \mid (b\ ;q),a))^*$. A state graph is



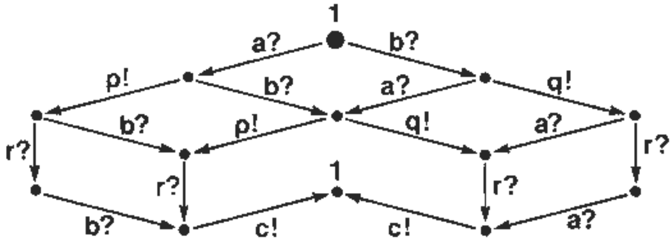
This component is a $\mathbf{C}_3$, the choice to be made being the one between outputs p and q. Notice that this specification does not exhibit a first come first serve principle. In delay-insensitive trace structures such a principle cannot be expressed. A realization of this component may exhibit a first come first serve behaviour, however.

(End of Example)

Example 2.10

In the arbiter of this example an additional symbol r is introduced that signals the reception by the environment of either p or q. Moreover, c is postponed until after the reception of r. For reasons explained in the next chapter we sometimes prefer this arbiter to the one in Example 2.9. The input alphabet of this component is $\{ a, b, r \}$, the output alphabet is $\{ c, p, q \}$, and the command is

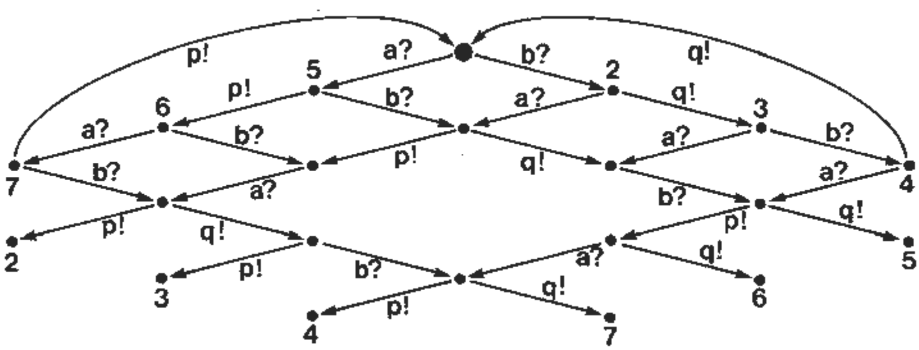
$(a,b,r;c)^*,((a;p;r),b \mid (b;q;r),a)^*$. A state graph, from which it can be seen that this component is a $\mathbf{C}_3$, is



(End of Example)

Example 2.11

The arbiter in this example allows multiple requests of one kind of symbol, e.g. a, without the need for the occurrence of the other symbol, b in this case. Its input alphabet is $\{a,b\}$ and its output alphabet $\{p,q\}$. A request, for a shared resource for example, is a high-going transition on one of the inputs a or b. A high-going transition on p means that request a has been granted and, similarly, a high-going transition on q that b has been granted. At most one request will be granted at a time. A low-going transition on the input whose request had been granted signals the release of the shared resource whereafter a low-going transition on the output that granted this request makes the arbiter ready for a next request of the same kind. The state graph, from which it can be seen that this component is a $\mathbf{C}_3$, is



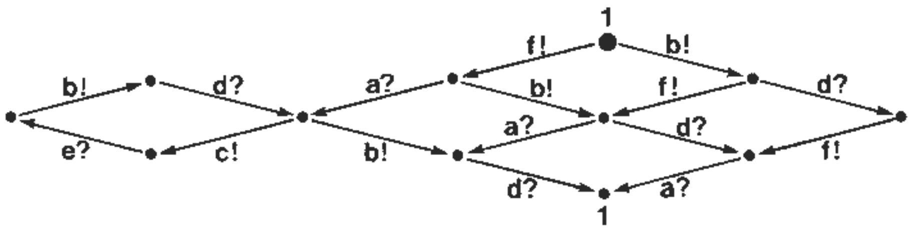
(End of Example)

Example 2.12

The component of this example is used to demonstrate that $\mathbf{C}_3$ is not closed under the composition operator to be introduced in the next chapter. It has input alphabet $\{a, d, e\}$, output alphabet $\{b, c, f\}$, and command

$$(((f\ ;a),(b\ ;d))^*\ ;f\ ;a\ ;(c\ ;e\ ;b\ ;d)^*\ ;b\ ;d)^*$$

A state graph of this component is



(End of Example)

We conclude this chapter with a number of lemmata. Lemmata 2.0 through 2.7 deal with a generalization of $\mathbf{R}_4''$. In Lemmata 2.8 through 2.11 we prove a few properties of $\mathbf{C}_2$'s in particular with respect to the shifting of output symbols to the right and input symbols to the left in traces of a $\mathbf{C}_2$.

Lemma 2.0 : For T a $\mathbf{C}_4$, for traces s and t, and for symbols a and b such that b is of another type than a and the symbols of t

$$sb \in \mathbf{t}T \wedge sabt \in \mathbf{t}T \Rightarrow sbat \in \mathbf{t}T$$

Proof : By mathematical induction on the length of t.

Base : $t = \epsilon$.

$sb \in \mathbf{t}T \wedge sabt \in \mathbf{t}T$

$\Rightarrow$ { $\mathbf{t}T$ is prefix-closed }

$sb \in \mathbf{t}T \wedge sa \in \mathbf{t}T$

$\Rightarrow$ { $\mathbf{R}_5'''$, using that a and b are of different types }

$sba \in \mathbf{t}T$

$=$ { $t = \epsilon$ }

$sbat \in \mathbf{t}T$

Step : $t = t_0c$. Hence, we have

b is of another type than c and the symbols of t_0 (0)

$sb \in \mathbf{t}\,T \wedge sabt \in \mathbf{t}\,T$
$= \{ t = t_0c$ and $\mathbf{t}\,T$ is prefix-closed $\}$
$sb \in \mathbf{t}\,T \wedge sabt_0 \in \mathbf{t}\,T \wedge sabt_0c \in \mathbf{t}\,T$
$\Rightarrow$ { induction hypothesis, using (0) }
$sbat_0 \in \mathbf{t}\,T \wedge sabt_0c \in \mathbf{t}\,T$
$\Rightarrow$ { $\mathbf{R}_4''$, using (0) }
$sbat_0c \in \mathbf{t}\,T$
$= \{ t = t_0c \}$
$sbat \in \mathbf{t}\,T$

(End of Proof)

Lemma 2.1 : For T a $\mathbf{C}_4$, for traces s and t, and for symbol b of another type than the symbols of t

$$sb \in \mathbf{t}\,T \wedge st \in \mathbf{t}\,T \Rightarrow sbt \in \mathbf{t}\,T$$

Proof : By mathematical induction on the length of t.

Base : $t = \epsilon$. Obvious.

Step : $t = at_0$. Hence, we have

b is of another type than a and the symbols of t_0 (0)

$sb \in \mathbf{t}\,T \wedge st \in \mathbf{t}\,T$
$= \{ t = at_0$ and $\mathbf{t}\,T$ is prefix-closed $\}$
$sb \in \mathbf{t}\,T \wedge sa \in \mathbf{t}\,T \wedge sat_0 \in \mathbf{t}\,T$
$\Rightarrow$ { $\mathbf{R}_5'''$, using (0) }
$sb \in \mathbf{t}\,T \wedge sab \in \mathbf{t}\,T \wedge sat_0 \in \mathbf{t}\,T$
$\Rightarrow$ { induction hypothesis, using (0) }
$sb \in \mathbf{t}\,T \wedge sabt_0 \in \mathbf{t}\,T$
$\Rightarrow$ { Lemma 2.0, using (0) }
$sbat_0 \in \mathbf{t}\,T$
$= \{ t = at_0 \}$

$sbt \in \mathbf{t}\,T$

(End of Proof)

Lemma 2.2 : For T a $\mathbf{C}_4$, for traces s and t, and for symbol b such that b is of another type than the symbols of t

$$sb \in \mathbf{t}\,T \wedge st \in \mathbf{t}\,T \Rightarrow (\forall w_0, w_1 : w_0 w_1 = t : sw_0 b w_1 \in \mathbf{t}\,T)$$

Proof : By mathematical induction on the length of t.

Base : $t = \epsilon$. Obvious.

Step : $t = at_0$. Hence, we have

$$b \text{ is of another type than } a \text{ and the symbols of } t_0 \qquad (0)$$

$sb \in \mathbf{t}\,T \wedge st \in \mathbf{t}\,T$

$=$ { Lemma 2.1. Moreover, $t = at_0$ and $\mathbf{t}\,T$ is prefix-closed }

$sbt \in \mathbf{t}\,T \wedge sb \in \mathbf{t}\,T \wedge sa \in \mathbf{t}\,T \wedge sat_0 \in \mathbf{t}\,T$

$\Rightarrow$ { $\mathbf{R}_5'''$, using (0) }

$sbt \in \mathbf{t}\,T \wedge sab \in \mathbf{t}\,T \wedge sat_0 \in \mathbf{t}\,T$

$\Rightarrow$ { induction hypothesis, using (0) }

$sbt \in \mathbf{t}\,T \wedge (\forall w_0, w_1 : w_0 w_1 = t_0 : saw_0 b w_1 \in \mathbf{t}\,T)$

$=$ { calculus }

$sbt \in \mathbf{t}\,T \wedge (\forall w_0, w_1 : aw_0 w_1 = at_0 : saw_0 b w_1 \in \mathbf{t}\,T)$

$=$ { $t = at_0$ and replacing aw_0 by w_0 }

$sbt \in \mathbf{t}\,T \wedge (\forall w_0, w_1 : w_0 w_1 = t \wedge w_0 \neq \epsilon : sw_0 b w_1 \in \mathbf{t}\,T)$

$=$ { calculus }

$(\forall w_0, w_1 : w_0 w_1 = t : sw_0 b w_1 \in \mathbf{t}\,T)$

(End of Proof)

Lemma 2.3 : For T a $\mathbf{C}_4$, for traces s, t, and u, and for symbols a and c such that a and c are of another type than the symbols of t

$$(\forall w_0, w_1 : w_0 w_1 = t : sw_0 a w_1 u \in \mathbf{t}\,T) \wedge satuc \in \mathbf{t}\,T$$
$$\Rightarrow (\forall w_0, w_1 : w_0 w_1 = t : sw_0 a w_1 uc \in \mathbf{t}\,T)$$

Proof : By mathematical induction on the length of t.

Base : $t = \epsilon$. Obvious.

Step : $t = bt_0$. Hence, we have

$$a \text{ and } c \text{ are of another type than } b \text{ and the symbols of } t_0 \qquad (0)$$

$(\forall w_0, w_1 : w_0w_1 = t : sw_0aw_1u \in \mathbf{t}\,T) \wedge satuc \in \mathbf{t}\,T$

$= \{ t = bt_0 \}$

$(\forall w_0, w_1 : w_0w_1 = bt_0 : sw_0aw_1u \in \mathbf{t}\,T) \wedge sabt_0uc \in \mathbf{t}\,T$

$\Rightarrow$ { calculus }

$(\forall w_0, w_1 : w_0w_1 = t_0 : sbw_0aw_1u \in \mathbf{t}\,T) \wedge sbat_0u \in \mathbf{t}\,T \wedge sabt_0uc \in \mathbf{t}\,T$

$\Rightarrow$ { $\mathbf{R}_4''$, using (0) }

$(\forall w_0, w_1 : w_0w_1 = t_0 : sbw_0aw_1u \in \mathbf{t}\,T) \wedge sbat_0uc \in \mathbf{t}\,T \wedge sabt_0uc \in \mathbf{t}\,T$

$\Rightarrow$ { induction hypothesis, using (0) }

$(\forall w_0, w_1 : w_0w_1 = t_0 : sbw_0aw_1uc \in \mathbf{t}\,T) \wedge sabt_0uc \in \mathbf{t}\,T$

$=$ { calculus }

$(\forall w_0, w_1 : w_0w_1 = bt_0 \wedge w_0 \neq \epsilon : sw_0aw_1uc \in \mathbf{t}\,T) \wedge sabt_0uc \in \mathbf{t}\,T$

$=$ { calculus and $t = bt_0$ }

$(\forall w_0, w_1 : w_0w_1 = t : sw_0aw_1uc \in \mathbf{t}\,T)$

(End of Proof)

In exactly the same way we derive

Lemma 2.4 : For T a $\mathbf{C}_4$, for traces s, t, and u, and for symbols b and c such that b is of another type than c and the symbols of t

$$(\forall w_0, w_1 : w_0w_1 = t : sw_0bw_1u \in \mathbf{t}\,T) \wedge stbuc \in \mathbf{t}\,T$$
$$\Rightarrow (\forall w_0, w_1 : w_0w_1 = t : sw_0bw_1uc \in \mathbf{t}\,T)$$

Lemma 2.5 : For T a $\mathbf{C}_4$, for traces s, t, and u, and for symbol a such that a is of another type than the symbols of t

$$satu \in \mathbf{t}\,T \wedge stau \in \mathbf{t}\,T \Rightarrow (\forall w_0, w_1 : w_0w_1 = t : sw_0aw_1u \in \mathbf{t}\,T)$$

Proof : By mathematical induction on the length of u.

Base : $u = \epsilon$.

$$
\begin{array}{ll}
 & satu \in \mathbf{t}T \wedge stau \in \mathbf{t}T \\
\Rightarrow & \{ \mathbf{t}T \text{ is prefix-closed} \} \\
 & sa \in \mathbf{t}T \wedge st \in \mathbf{t}T \\
\Rightarrow & \{ \text{Lemma 2.2, since } a \text{ is of another type than the symbols of } t \} \\
 & (\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 \in \mathbf{t}T) \\
\Rightarrow & \{ u = \epsilon \} \\
 & (\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 u \in \mathbf{t}T)
\end{array}
$$

Step : $u = u_0 b$.

$$
\begin{array}{ll}
 & satu \in \mathbf{t}T \wedge stau \in \mathbf{t}T \\
= & \{ u = u_0 b \text{ and } \mathbf{t}T \text{ is prefix-closed} \} \\
 & satu_0 \in \mathbf{t}T \wedge stau_0 \in \mathbf{t}T \wedge satu_0 b \in \mathbf{t}T \wedge stau_0 b \in \mathbf{t}T \\
\Rightarrow & \{ \text{induction hypothesis} \} \\
 & (\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 u_0 \in \mathbf{t}T) \wedge satu_0 b \in \mathbf{t}T \wedge stau_0 b \in \mathbf{t}T \\
\Rightarrow & \{ \text{Lemma 2.3 if the types of } a \text{ and } b \text{ are equal, Lemma 2.4 if they are not} \} \\
 & (\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 u_0 b \in \mathbf{t}T) \\
= & \{ u = u_0 b \} \\
 & (\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 u \in \mathbf{t}T)
\end{array}
$$

(End of Proof)

Lemma 2.6 : For T a $\mathbf{C}_4$, for traces s, t, and u, and for symbols a and c such that the symbols of t are of another type than a and c

$$satuc \in \mathbf{t}T \wedge stau \in \mathbf{t}T \Rightarrow stauc \in \mathbf{t}T$$

Proof:

$$
\begin{array}{ll}
 & satuc \in \mathbf{t}T \wedge stau \in \mathbf{t}T \\
= & \{ \mathbf{t}T \text{ is prefix-closed} \} \\
 & satu \in \mathbf{t}T \wedge stau \in \mathbf{t}T \wedge satuc \in \mathbf{t}T \\
\Rightarrow & \{ \text{Lemma 2.5} \} \\
 & (\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 u \in \mathbf{t}T) \wedge satuc \in \mathbf{t}T \\
\Rightarrow & \{ \text{Lemma 2.3} \} \\
 & (\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 uc \in \mathbf{t}T)
\end{array}
$$

$\Rightarrow$ { instantiation }

$stauc \in \mathbf{t}\,T$

(End of Proof)

In a similar way, applying Lemma 2.4 instead of 2.3, we derive

Lemma 2.7 : For T a $\mathbf{C}_4$, for traces s, t, and u, and for symbols b and c such that b is of another type than c and the symbols of t

$$stbuc \in \mathbf{t}\,T \ \wedge\ sbtu \in \mathbf{t}\,T \Rightarrow sbtuc \in \mathbf{t}\,T$$

Finally we prove a few lemmata on the shifting of symbols in $\mathbf{C}_2$'s.

Lemma 2.8 : For T a $\mathbf{C}_2$, for traces s and t, and for symbol $a \in \mathbf{o}\,T$ such that $t\lceil\{\,a\,\} = \epsilon$

$$sa \in \mathbf{t}\,T \ \wedge\ st \in \mathbf{t}\,T \Rightarrow sta \in \mathbf{t}\,T$$

Proof : By mathematical induction on the length of t.

Base : $t = \epsilon$. Obvious.

Step : $t = t_0 b$. Hence, we have

$$t_0\lceil\{\,a\,\} = \epsilon \ \text{ and } \ a \neq b \qquad (0)$$

$sa \in \mathbf{t}\,T \ \wedge\ st \in \mathbf{t}\,T$

$=$ { $t = t_0 b$ and $\mathbf{t}\,T$ is prefix-closed }

$sa \in \mathbf{t}\,T \ \wedge\ st_0 \in \mathbf{t}\,T \ \wedge\ st_0 b \in \mathbf{t}\,T$

$\Rightarrow$ { induction hypothesis, using (0) }

$st_0 a \in \mathbf{t}\,T \ \wedge\ st_0 b \in \mathbf{t}\,T$

$\Rightarrow$ { $\mathbf{R}_5''$, using $a \in \mathbf{o}\,T$ and $a \neq b$ according to (0) }

$st_0 ba \in \mathbf{t}\,T$

$=$ { $t = t_0 b$ }

$sta \in \mathbf{t}\,T$

(End of Proof)

Lemma 2.9 : For T a $\mathbf{C}_2$, for traces s, t, and u, and for symbol $a \in \mathbf{o}\,T$ such that $t \lceil \{ a \} = \epsilon$

$$sa \in \mathbf{t}\,T \wedge stau \in \mathbf{t}\,T \Rightarrow satu \in \mathbf{t}\,T$$

Proof : By mathematical induction on the length of t.

Base : $t = \epsilon$. Obvious.

Step : $t = t_0 b$. Hence, we have

$$t_0 \lceil \{ a \} = \epsilon \text{ and } a \neq b \tag{0}$$

$sa \in \mathbf{t}\,T \wedge stau \in \mathbf{t}\,T$

$=$ { $t = t_0 b$ and $\mathbf{t}\,T$ is prefix-closed }

$sa \in \mathbf{t}\,T \wedge st_0 \in \mathbf{t}\,T \wedge st_0 bau \in \mathbf{t}\,T$

$\Rightarrow$ { Lemma 2.8, using (0) and $a \in \mathbf{o}\,T$ }

$sa \in \mathbf{t}\,T \wedge st_0 a \in \mathbf{t}\,T \wedge st_0 bau \in \mathbf{t}\,T$

$\Rightarrow$ { $\mathbf{R}_4'$ if a and b are of different types, $\mathbf{R}_3$ if they are of the same type }

$sa \in \mathbf{t}\,T \wedge st_0 abu \in \mathbf{t}\,T$

$\Rightarrow$ { induction hypothesis, using (0) }

$sat_0 bu \in \mathbf{t}\,T$

$=$ { $t = t_0 b$ }

$satu \in \mathbf{t}\,T$

(End of Proof)

Lemma 2.10 : For T a $\mathbf{C}_2$, for traces s, t, and u, and for symbol $a \in \mathbf{o}\,T$ such that $t \lceil \{ a \} = \epsilon$

$$sa \in \mathbf{t}\,T \wedge stau \in \mathbf{t}\,T \Rightarrow (\forall w_0, w_1 : w_0 w_1 = t : sw_0 aw_1 u \in \mathbf{t}\,T)$$

Proof :

$sa \in \mathbf{t}\,T \wedge stau \in \mathbf{t}\,T$

$=$ { $\mathbf{t}\,T$ is prefix-closed and calculus }

$(\forall w_0, w_1 : w_0 w_1 = t : sa \in \mathbf{t}\,T \wedge sw_0 \in \mathbf{t}\,T \wedge sw_0 w_1 au \in \mathbf{t}\,T)$

$\Rightarrow$ { since $t \lceil \{ a \} = \epsilon$, we have, if $w_0 w_1 = t$, $w_0 \lceil \{ a \} = \epsilon$. Hence, we may apply Lemma 2.8 }

$(\forall w_0, w_1 : w_0 w_1 = t : sw_0 a \in \mathbf{t}\,T \wedge sw_0 w_1 au \in \mathbf{t}\,T)$

$\Rightarrow$ { Lemma 2.9 }

$(\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 u \in \mathbf{t}\, T)$

(End of Proof)

Lemma 2.11 : For T a $\mathbf{C}_2$, for traces s, t, and u, and for symbol $a \in \mathbf{i}\, T$

$$(\forall w_0, w_1 : w_0 w_1 = t : s w_0 a \in \mathbf{t}\, T) \wedge stau \in \mathbf{t}\, T$$
$$\Rightarrow (\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 u \in \mathbf{t}\, T)$$

Proof : By mathematical induction on the length of t.

Base : $t = \epsilon$. Straightforward.

Step : $t = t_0 b$. Then we derive

$(\forall w_0, w_1 : w_0 w_1 = t : s w_0 a \in \mathbf{t}\, T) \wedge stau \in \mathbf{t}\, T$

$=$ { $t = t_0 b$ }

$(\forall w_0, w_1 : w_0 w_1 = t_0 b : s w_0 a \in \mathbf{t}\, T) \wedge s t_0 b a u \in \mathbf{t}\, T$

$\Rightarrow$ { calculus }

$(\forall w_0, w_1 : w_0 w_1 = t_0 : s w_0 a \in \mathbf{t}\, T) \wedge s t_0 a \in \mathbf{t}\, T \wedge s t_0 b a u \in \mathbf{t}\, T$

$\Rightarrow$ { $\mathbf{R}_3$ if a and b are of the same type. $\mathbf{R}_4{}'$ if they are of different types }

$(\forall w_0, w_1 : w_0 w_1 = t_0 : s w_0 a \in \mathbf{t}\, T) \wedge s t_0 a b u \in \mathbf{t}\, T \wedge s t_0 b a u \in \mathbf{t}\, T$

$\Rightarrow$ { induction hypothesis }

$(\forall w_0, w_1 : w_0 w_1 = t_0 : s w_0 a w_1 b u \in \mathbf{t}\, T) \wedge s t_0 b a u \in \mathbf{t}\, T$

$=$ { calculus }

$(\forall w_0, w_1 : w_0 w_1 = t_0 b \wedge w_1 \neq \epsilon : s w_0 a w_1 u \in \mathbf{t}\, T) \wedge s t_0 b a u \in \mathbf{t}\, T$

$=$ { calculus and $t = t_0 b$ }

$(\forall w_0, w_1 : w_0 w_1 = t : s w_0 a w_1 u \in \mathbf{t}\, T)$

(End of Proof)

3

Independent alphabets and composition

In this chapter we introduce so-called independent alphabets. Informally speaking, we partition the environment of a component in such a way that the subenvironments are mutually independent with respect to their communications with that component. Such a partitioning is, for example, a justification for sometimes conceiving the environment as being divided into a left and a right environment. In the last section a composition operator is defined using independent alphabets.

3.0. Independent alphabets

Outputs of the component are under control of the component and inputs of the component are under control of the environment. The component will operate according to its specification by sending outputs as long as the environment sends outputs that the component is able to receive according to that specification, in other words as long as there is absence of computation interference.

Composition of two electrical circuits usually involves the interconnection of just a subset of wires of the circuits to be composed. Communications via these wires are the composite's internal communications. The remaining wires are used for the external communications, i.e. the communications of the composite with its environment. Therefore, the environment of each component is partitioned, upon composition, into an environment for the internal and an environment for the external communications. This implies two so-called local specifications, viz. the one obtained by projecting the original specification onto the symbols used for the internal communications and the one obtained by projecting onto the symbols used for the external communications. A nice property of this partitioning would be that the internal and external communications could be carried out according to the rules of the preceding paragraph just with respect to their local specifications, i.e. by locally guaranteeing absence of computation interference guaranteeing absence of computation interference for the whole. This is captured in the requirement that if an input symbol is allowed to occur according to a local specification then it is also allowed to occur according to the global one. Formally this is defined as follows.

Definition 3.0 : For T a C_4, alphabet C, $C \subseteq \mathbf{a}\,T$, is independent with respect to T if

$$(\forall s,a : s \in \mathbf{t}\,T \wedge a \in C \cap \mathbf{i}\,T : sa\lceil C \in \mathbf{t}\,T\lceil C = sa \in \mathbf{t}\,T) \wedge$$
$$(\forall s,a : s \in \mathbf{t}\,T \wedge a \in \overline{C} \cap \mathbf{i}\,T : sa\lceil \overline{C} \in \mathbf{t}\,T\lceil \overline{C} = sa \in \mathbf{t}\,T)$$

where the complement of C with respect to $\mathbf{a}\,T$ is denoted by $\overline{C}$.
(End of Definition)

Notice that $\mathbf{a}\,T$ itself is independent with respect to trace structure T. The equality could be replaced by an implication since $sa \in \mathbf{t}\,T \Rightarrow sa\lceil C \in \mathbf{t}\,T\lceil C$ by definition. Moreover, it can be seen that independence of C is the same as independence of $\overline{C}$.

One of the requirements for composition of two components will be that their set of common symbols be independent with respect to both components. This is sufficient to guarantee absence of computation interference as far as external input symbols are concerned, as will be proved in Lemma 3.5. Additional requirements are needed to guarantee absence of computation interference for the internal inputs. First, however, we illustrate the definition of the notion of independent alphabet using some examples of the preceding chapter.

Example 3.0

Consider a C-element with two output wires as in Example 2.2. The input alphabet is $\{a,b\}$, the output alphabet is $\{c,d\}$. The component with command $a,b\,;((c\,;a),(d\,;b))^*$ has independent alphabets $\{a,c\}$ and $\{b,d\}$. Projection on $\{a,c\}$ yields a trace structure with command $(a\,;c)^*$. The traces in this trace structure that contain an equal number of a's and c's may be extended with a. Traces of the original trace structure with an equal number of a's and c's may be extended with a as well, as can easily be seen from the command. For reasons of symmetry, something similar holds for alphabet $\{b,d\}$.
Taking the component with command $(a,b\,;c,d)^*$, however, one cannot find independent alphabets other than the trivial ones. Trace abc in this trace structure, for instance, cannot be extended with a, although its projection on $\{a,c\}$,

being ac, may be extended with a in the projection of the trace structure onto $\{a,c\}$.

(End of Example)

Example 3.1

The C-wire element of Example 2.3 with input alphabet $\{a,q,r\}$, output alphabet $\{b,p\}$ and command $a;(p;(q;b;a),r)^*$ has independent alphabets $\{a,b\}$ and $\{p,q,r\}$. Projection on $\{a,b\}$ yields a trace structure with command $(a;b)^*$. As in the preceding example, the traces of this trace structure that contain an equal number of a's and b's may be extended with input a. The same holds for the traces of the original trace structure as can be seen from the command. Consequently, with respect to alphabet $\{a,b\}$ the first of the two conditions of independence is met. Moreover, projection on $\{p,q,r\}$ yields a trace structure with command $(p;q,r)^*$ with output p and inputs q and r. The traces of this trace structure that have a lead of p over q may he extended with q and traces that have a lead of p over r may be extended with r. The same holds for the traces in the original trace structure.

(End of Example)

Example 3.2

The three wires of Example 2.4 have independent alphabets as well. The input alphabet is $\{x_0,x_1,b\}$, the output alphabet is $\{y_0,y_1,a\}$, and the command is $(x_0;y_0;b;a \mid x_1;y_1;b;a)^*$. The alphabets $\{x_0,x_1,a\}$ and $\{y_0,y_1,b\}$ are independent. Symbol a may immediately be followed by either x_0 or x_1, and symbols y_0 or y_1 by b. Notice that x_0 and x_1, which are two input symbols that disable one another, necessarily belong to the same independent alphabet. This is one of the reasons that the partitioning into the three wires $\{x_0,y_0\}$, $\{x_1,y_1\}$, and $\{a,b\}$ does not yield independent alphabets. Notice also that, although the component is a $\mathbf{C}_2$, the projection on independent alphahet $\{y_0,y_1,b\}$ is a $\mathbf{C}_3$. Nevertheless, we prove in Chapter 5 that composing two $\mathbf{C}_2$'s, using the composition operator that is defined in the next section, yields a $\mathbf{C}_2$ again.

(End of Example)

Example 3.3

Consider the And-element of Example 2.6 with input alphabet $\{a,r\}$, output alphahet $\{b,c,p\}$, and command $(a;p;r;b,c;a;c,(p;r;b))^*$. It has independent alphabets $\{a,b,c\}$ and $\{p,r\}$. In the trace structure that results after projection on $\{a,b,c\}$, having command $(a;b,c)^*$, the traces with an equal number of a's, b's, and c's may be extended with input a. The same holds for traces with this property in the original trace structure. For alphabet $\{p,r\}$ it is even more clear that the requirements of independence are met.

(End of Example)

Example 3.4

The buffer of Example 2.8 has been constructed in such a way that data storage and data retrieval can be performed simultaneously. For data storage x_0 and x_1 are used and the request for new data is passed by a. Outputs y_0 and y_1 return the stored value on request b. Indeed, alphabets $\{x_0,x_1,a\}$ and $\{y_0,y_1,b\}$ are independent as can be seen from the state graph. After a either x_0 or x_1 is possible both in the original trace structure and in the trace structure with command $(x_0 \mid x_1 ; a)^*$, which results after projection on $\{x_0,x_1,a\}$. Projection on $\{y_0,y_1,b\}$ yields command $(y_0 \mid y_1 ; b)^*$. After y_0 or y_1, both in this and in the original trace structure b is possible.

(End of Example)

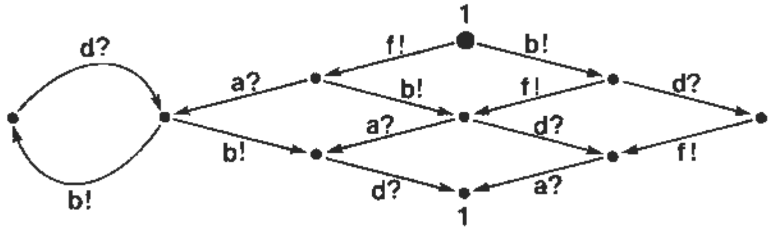
Example 3.5

The reason that the arbiter of Example 2.10 is sometimes preferred to the one of Example 2.9 is that the former's alphabet can be partitioned into independent alphabets. The input alphabet is $\{a,b,r\}$, the output alphabet $\{c,p,q\}$, and the command $(a,b,r;c)^*,((a;p;r),b \mid (b;q;r),a)^*$. Independent alphabets are $\{a,b,c\}$ and $\{p,q,r\}$. Projection on $\{a,b,c\}$ yields command $(a,b;c)^*$, from which we infer that the traces in this trace structure that have an equal number of a's, b's, and c's may be extended with a and b in either order. The traces in the original trace structure have the same property. Projection on $\{p,q,r\}$ yields command $((p \mid q);r)^*$, where p and q are outputs and r is an input. A trace in this trace structure may be extended with r if the sum of the numbers of p's and q's exceeds the number of r's in that trace. The original trace structure has the same property.

(End of Example)

Example 3.6

The component of Example 2.12 has independent alphabets $\{c,e\}$ and $\{a,b,d,f\}$. Notice that, as opposed to inputs, outputs that disable one another may belong to different independent alphabets (to which the fact that $\mathbf{C}_3$ is not closed under composition can be attributed). The state graph that results after projection on $\{a,b,d,f\}$ is



Notice that this trace structure does not satisfy $\mathbf{R}_4'$ anymore. Traces *fb* and *fabdbdb* belong to the trace structure, whereas *fbadbdb* does not. Notice also that $\{a, f\}$ and $\{b, d\}$ are independent alphabets with respect to this trace structure. Projection on such an alphabet, however, yields a $\mathbf{C}_1$ again.

(End of Example)

We conclude this section with a number of lemmata. We show that an input symbol of an independent alphabet C may be shifted to the left over symbols of $\bar{C}$ (and similarly output symbols to the right). Moreover, we prove that a $\mathbf{C}_4$ projected on an independent alphabet is a $\mathbf{C}_4$ again.

Lemma 3.0 : For T a $\mathbf{C}_4$ with independent alphabet C, for traces s and t, and for symbols $a \in \mathbf{a}\,T$ and $b \in C \cap \mathbf{i}\,T$

$$sabt \in \mathbf{t}\,T \ \wedge\ sbat \lceil C \in \mathbf{t}\,T \lceil C \ \Rightarrow\ sbat \in \mathbf{t}\,T$$

Proof : If $a \in \mathbf{i}\,T$ this lemma is a consequence of $\mathbf{R}_3$. Therefore, assume

$$a \in \mathbf{o}\,T \tag{0}$$

We prove the lemma by mathematical induction on the length of t.

Base : $t = \epsilon$.

$sabt \in \mathbf{t}\,T \ \wedge\ sbat \lceil C \in \mathbf{t}\,T \lceil C$

$\Rightarrow$ { $\mathbf{t}\,T$ is prefix-closed and so is $\mathbf{t}\,T \lceil C$ according to Property 1.2 }

$sa \in \mathbf{t}\,T \ \wedge\ s \in \mathbf{t}\,T \ \wedge\ sb \lceil C \in \mathbf{t}\,T \lceil C$

$\Rightarrow$ { C is independent with respect to T and $b \in C \cap \mathbf{i}\,T$ }

$sa \in \mathbf{t}\,T \ \wedge\ sb \in \mathbf{t}\,T$

$\Rightarrow$ { $\mathbf{R}_5'''$, using $b \in \mathbf{i}\,T$ and $a \in \mathbf{o}\,T$ according to (0) }

$sba \in \mathbf{t}\,T$

$=$ { $t = \epsilon$ }

$sbat \in \mathbf{t}\,T$

Step : $t = t_0c$. Assume the left-hand side of the implication. Hence,

$$sabt \in \mathbf{t}\,T \quad \text{and} \quad sbat \lceil C \in \mathbf{t}\,T \lceil C \tag{1}$$

Then we derive

$$\begin{array}{ll} & \text{true} \\ = & \{ \text{(1), using } t = t_0c \text{ and the prefix-closedness of } \mathbf{t}\,T \text{ and } \mathbf{t}\,T \lceil C \ \} \\ & sabt_0 \in \mathbf{t}\,T \ \wedge \ sbat_0 \lceil C \in \mathbf{t}\,T \lceil C \\ \Rightarrow & \{ \text{ induction hypothesis } \} \\ & sbat_0 \in \mathbf{t}\,T \end{array} \tag{2}$$

Next, we distinguish three cases : (i) $c \in \mathbf{o}\,T$, (ii) $c \in C \cap \mathbf{i}\,T$, and (iii) $c \in \mathbf{i}\,T \setminus C$. We prove that $sbat_0c \in \mathbf{t}\,T$ which yields the result desired, since $t = t_0c$.

(i) $c \in \mathbf{o}\,T$

$$\begin{array}{ll} & \text{true} \\ = & \{ \text{ (1) and (2), using } t = t_0c \ \} \\ & sabt_0c \in \mathbf{t}\,T \ \wedge \ sbat_0 \in \mathbf{t}\,T \\ \Rightarrow & \{ \ \mathbf{R}_4'', \text{ since } b \in \mathbf{i}\,T,\ a \in \mathbf{o}\,T \text{ according to (0), and } c \in \mathbf{o}\,T \ \} \\ & sbat_0c \in \mathbf{t}\,T \end{array}$$

(ii) $c \in C \cap \mathbf{i}\,T$

$$\begin{array}{ll} & \text{true} \\ = & \{ \text{ (1) and (2), using } t = t_0c \ \} \\ & sbat_0c \lceil C \in \mathbf{t}\,T \lceil C \ \wedge \ sbat_0 \in \mathbf{t}\,T \\ \Rightarrow & \{ \ C \text{ is independent with respect to } T \text{ and } c \in C \cap \mathbf{i}\,T \ \} \\ & sbat_0c \in \mathbf{t}\,T \end{array}$$

(iii) $c \in \mathbf{i}\,T \setminus C$

$$\begin{array}{ll} & \text{true} \\ = & \{ \text{ (1), using } t = t_0c \text{ and projection on } \mathbf{a}\,T \setminus C \text{, and (2) } \} \\ & sabt_0c \lceil (\mathbf{a}\,T \setminus C) \in \mathbf{t}\,T \lceil (\mathbf{a}\,T \setminus C) \ \wedge \ sbat_0 \in \mathbf{t}\,T \end{array}$$

$=$ { distribution of projection over concatenation, using $b \in C$ }

$sbat_0c\lceil(\mathbf{a}\,T \setminus C) \in \mathbf{t}\,T\lceil(\mathbf{a}\,T \setminus C) \wedge sbat_0 \in \mathbf{t}\,T$

$\Rightarrow$ { $\mathbf{a}\,T \setminus C$ is independent with respect to T, since C is, and $c \in (\mathbf{a}\,T \setminus C) \cap \mathbf{i}\,T$ }

$sbat_0c \in \mathbf{t}\,T$

(End of Proof)

Lemma 3.1 : For T a $\mathbf{C}_4$ with independent alphabet C, for traces s, t, and u, and for symbol $a \in C \cap \mathbf{i}\,T$ such that $t\lceil C = \epsilon$

$$stau \in \mathbf{t}\,T \Rightarrow satu \in \mathbf{t}\,T$$

Proof : By mathematical induction on the length of t.

Base : $t = \epsilon$. Obvious.

Step : $t = t_0b$. Hence, we have

$$t_0\lceil C = \epsilon \text{ and } b \notin C \qquad (0)$$

$stau \in \mathbf{t}\,T$

$=$ { $t = t_0b$ and projection on C }

$st_0bau \in \mathbf{t}\,T \wedge st_0bau\lceil C \in \mathbf{t}\,T\lceil C$

$=$ { distribution of projection over concatenation, using $b \notin C$ according to (0) }

$st_0bau \in \mathbf{t}\,T \wedge st_0abu\lceil C \in \mathbf{t}\,T\lceil C$

$\Rightarrow$ { Lemma 3.0, since $a \in C \cap \mathbf{i}\,T$ }

$st_0abu \in \mathbf{t}\,T$

$\Rightarrow$ { induction hypothesis, using (0) }

$sat_0bu \in \mathbf{t}\,T$

$=$ { $t = t_0b$ }

$satu \in \mathbf{t}\,T$

(End of Proof)

In a similar way, using that $\mathbf{a}\,T \setminus C$ is independent as well, we derive

Lemma 3.2 : For T a $\mathbf{C}_4$ with independent alphabet C, for traces s, t, and u, and for symbol $a \in C \cap \mathbf{o}\,T$ such that $t \lceil C = \epsilon$

$$satu \in \mathbf{t}\,T \Rightarrow stau \in \mathbf{t}\,T$$

Often we only want two symbols of the same type to be adjacent and we are not interested in the direction of the shifting. Therefore, we combine the last two lemmata, which yields

Lemma 3.3 : For T a $\mathbf{C}_4$ with independent alphabet C, for traces s, t, and u, and for symbols $a \in C$ and $b \in C$ of the same type such that $t \lceil C = \epsilon$

$$satbu \in \mathbf{t}\,T \Rightarrow sabtu \in \mathbf{t}\,T \ \vee \ stabu \in \mathbf{t}\,T$$

Lemma 3.4 : For T a $\mathbf{C}_4$ with independent alphabet C, $T \lceil C$ is a $\mathbf{C}_4$ again.

Proof : We have to prove the 6 rules of the definition of $\mathbf{C}_4$ to hold for $T \lceil C$. $\mathbf{R}_0$ through $\mathbf{R}_3$ are fairly easy to prove, using for $\mathbf{R}_2$ and $\mathbf{R}_3$ Lemma 3.3. We prove $\mathbf{R}_4''$ and $\mathbf{R}_5'''$ only.

$\mathbf{R}_4''$: for traces s and t, and for symbols $a \in C$, $b \in C$, and $c \in C$ such that b is of another type than a and c

$$sabtc \in \mathbf{t}\,T \lceil C \ \wedge \ sbat \in \mathbf{t}\,T \lceil C \Rightarrow sbatc \in \mathbf{t}\,T \lceil C$$

We distinguish two cases : (i) $b \in \mathbf{i}\,T$, and (ii) $b \in \mathbf{o}\,T$

(i) $b \in \mathbf{i}\,T$

$$
\begin{array}{ll}
 & sabtc \in \mathbf{t}\,T \lceil C \ \wedge \ sbat \in \mathbf{t}\,T \lceil C \\
= & \{ \text{ definition of projection, using that } \mathbf{t}\,T \text{ is prefix-closed } \} \\
 & (\exists s_0, s_1, s_2 :: s_0 a s_1 b s_2 c \in \mathbf{t}\,T \ \wedge \ sbat \in \mathbf{t}\,T \lceil C \\
 & \qquad\qquad \wedge \ s_0 \lceil C = s \ \wedge \ s_1 \lceil C = \epsilon \ \wedge \ s_2 \lceil C = t) \\
\Rightarrow & \{ \text{ Lemma 3.1, since } C \text{ is independent with respect to } T \text{, renaming } \} \\
 & (\exists s_0, s_1 :: s_0 ab s_1 c \in \mathbf{t}\,T \ \wedge \ sbat \in \mathbf{t}\,T \lceil C \ \wedge \ s_0 \lceil C = s \ \wedge \ s_1 \lceil C = t) \\
= & \{ \text{ calculus, } \mathbf{t}\,T \text{ is prefix-closed, and distribution of projection over concatenation, using } a \in C \text{ and } b \in C \ \}
\end{array}
$$

$(\exists s_0, s_1 :: s_0 abs_1 c \in \mathbf{t}\, T \ \wedge\ s_0 abs_1 \in \mathbf{t}\, T \ \wedge\ s_0 bas_1 \lceil C \in \mathbf{t}\, T \lceil C$
$\qquad \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = t)$

$\Rightarrow$ { Lemma 3.0, since $b \in C \cap \mathbf{i}\, T$ }

$(\exists s_0, s_1 :: s_0 abs_1 c \in \mathbf{t}\, T \ \wedge\ s_0 bas_1 \in \mathbf{t}\, T \ \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = t)$

$\Rightarrow$ { $\mathbf{R}_4''$ }

$(\exists s_0, s_1 :: s_0 bas_1 c \in \mathbf{t}\, T \ \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = t)$

$\Rightarrow$ { projection, using $a \in C$, $b \in C$, and $c \in C$ }

$sbatc \in \mathbf{t}\, T \lceil C$

(ii) $b \in \mathbf{o}\, T$

$sabtc \in \mathbf{t}\, T \lceil C \ \wedge\ sbat \in \mathbf{t}\, T \lceil C$

$=$ { definition of projection }

$(\exists s_0, s_1, s_2 :: sabtc \in \mathbf{t}\, T \lceil C \ \wedge\ s_0 bs_1 as_2 \in \mathbf{t}\, T$
$\qquad \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = \epsilon \ \wedge\ s_2 \lceil C = t)$

$\Rightarrow$ { Lemma 3.2, since C is independent with respect to T, renaming }

$(\exists s_0, s_1 :: s_0 bas_1 \in \mathbf{t}\, T \ \wedge\ sabtc \in \mathbf{t}\, T \lceil C \ \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = t)$

$=$ { calculus, $\mathbf{t}\, T \lceil C$ is prefix-closed, and distribution of projection over concatenation, using $a \in C$, $b \in C$, and $c \in C$ }

$(\exists s_0, s_1 :: s_0 bas_1 \in \mathbf{t}\, T \ \wedge\ s_0 abs_1 \lceil C \in \mathbf{t}\, T \lceil C \ \wedge\ s_0 abs_1 c \lceil C \in \mathbf{t}\, T \lceil C$
$\qquad \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = t)$

$\Rightarrow$ { Lemma 3.0, since a is of another type than b and, hence, $a \in C \cap \mathbf{i}\, T$ }

$(\exists s_0, s_1 :: s_0 bas_1 \in \mathbf{t}\, T \ \wedge\ s_0 abs_1 \in \mathbf{t}\, T \ \wedge\ s_0 abs_1 c \lceil C \in \mathbf{t}\, T \lceil C$
$\qquad \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = t)$

$\Rightarrow$ { C is independent with respect to T and c is of another type than b and, hence, $c \in C \cap \mathbf{i}\, T$ }

$(\exists s_0, s_1 :: s_0 bas_1 \in \mathbf{t}\, T \ \wedge\ s_0 abs_1 c \in \mathbf{t}\, T \ \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = t)$

$\Rightarrow$ { $\mathbf{R}_4''$ }

$(\exists s_0, s_1 :: s_0 bas_1 c \in \mathbf{t}\, T \ \wedge\ s_0 \lceil C = s \ \wedge\ s_1 \lceil C = t)$

$\Rightarrow$ { projection, using $a \in C$, $b \in C$, and $c \in C$ }

$sbatc \in \mathbf{t}\, T \lceil C$

$\mathbf{R}_5'''$: for trace s, and for symbols $a \in C$ and $b \in C$ of different types

$$sa \in \mathbf{t}\, T \lceil C \ \wedge\ sb \in \mathbf{t}\, T \lceil C \Rightarrow sab \in \mathbf{t}\, T \lceil C$$

Assuming that $a \in C \cap \mathbf{i}\,T$ and $b \in C \cap \mathbf{o}\,T$ we derive

$$
\begin{array}{ll}
& sa \in \mathbf{t}\,T\lceil C \;\wedge\; sb \in \mathbf{t}\,T\lceil C \\
= & \{\text{ definition of projection, using } \mathbf{t}\,T \text{ is prefix-closed and } b \in C \;\} \\
& (\exists s_0 :: sa \in \mathbf{t}\,T\lceil C \;\wedge\; s_0 \in \mathbf{t}\,T \;\wedge\; s_0 b \in \mathbf{t}\,T \;\wedge\; s_0\lceil C = s) \\
= & \{\text{ calculus and distribution of projection over concatenation, using } a \in C \;\} \\
& (\exists s_0 :: s_0 a\lceil C \in \mathbf{t}\,T\lceil C \;\wedge\; s_0 \in \mathbf{t}\,T \;\wedge\; s_0 b \in \mathbf{t}\,T \;\wedge\; s_0\lceil C = s) \\
\Rightarrow & \{\; C \text{ is independent with respect to } T \text{ and } a \in C \cap \mathbf{i}\,T \;\} \\
& (\exists s_0 :: s_0 a \in \mathbf{t}\,T \;\wedge\; s_0 b \in \mathbf{t}\,T \;\wedge\; s_0\lceil C = s) \\
\Rightarrow & \{\; \mathbf{R}_5''' \;\} \\
& (\exists s_0 :: s_0 ab \in \mathbf{t}\,T \;\wedge\; s_0 ba \in \mathbf{t}\,T \;\wedge\; s_0\lceil C = s) \\
\Rightarrow & \{\text{ projection, using } a \in C \text{ and } b \in C \;\} \\
& sab \in \mathbf{t}\,T\lceil C \;\wedge\; sba \in \mathbf{t}\,T\lceil C
\end{array}
$$

Hence, $\mathbf{R}_5'''$ holds for $a \in C \cap \mathbf{o}\,T$ and $b \in C \cap \mathbf{i}\,T$ as well.

(End of Proof)

3.1. Composition

Using independent alphabets we can state a number of conditions that guarantee absence of interference when composing two delay-insensitive trace structures. As we have argued in the preceding section and as will be proved in Lemma 3.5, blending two components by means of a set of common symbols that is independent with respect to both components guarantees absence of computation interference for the external communications with respect to the internal ones. In addition we impose two restrictions upon the internal communications. The first one is that each common symbol be an output symbol of the one and an input symbol of the other component. Second, we require the projections of both specifications on the set of common symbols to be equal. Formally this is captured in the following way.

Definition 3.1 : Two $\mathbf{C}_4$'s S and T are connectable if

0) $\mathbf{a}S \cap \mathbf{a}T = (\mathbf{o}S \cap \mathbf{i}\,T) \cup (\mathbf{i}S \cap \mathbf{o}T)$

1) $\mathbf{a}S \cap \mathbf{a}T$ is independent with respect to both S and T

2) $S\lceil(\mathbf{a}S \cap \mathbf{a}T) = T\lceil(\mathbf{a}S \cap \mathbf{a}T)$

(End of Definition)

Notice that requirement 0) indeed states that each common symbol is an output symbol in the one and an input symbol in the other component, since $\mathbf{i}S \cap \mathbf{o}S = \varnothing$ (and $\mathbf{i}T \cap \mathbf{o}T = \varnothing$). Requirement 2) in particular is a very stringent one. Even under these restrictions, however, it turns out to be quite delicate to prove absence of computation and transmission interference for the internal communications or to prove the various closure properties. Therefore, we confine ourselves in this monograph to Definition 3.1, indicating in Chapter 6 a number of ways to relax requirements 1) and 2).

As a preparation of the proof of absence of computation and transmission interference we conclude this section stating a few properties with regard to the input and output alphabets of the blend of two connectable $\mathbf{C}_4$'s. They may be proved using that

$$\mathbf{o}(S\ \mathbf{w}\ T) = \mathbf{o}(S\ \mathbf{b}\ T) = (\mathbf{o}S \cup \mathbf{o}T) \setminus (\mathbf{a}S \cap \mathbf{a}T) \text{ and}$$
$$\mathbf{i}(S\ \mathbf{w}\ T) = \mathbf{i}(S\ \mathbf{b}\ T) = (\mathbf{i}S \cup \mathbf{i}T) \setminus (\mathbf{a}S \cap \mathbf{a}T)$$

according to Chapter 1, and that the alphabet of a $\mathbf{C}_4$ consists of input and output symbols only according to $\mathbf{R}_0$.

Property 3.0 : For connectable $\mathbf{C}_4$'s S and T

(i) $\mathbf{i}S \cap \mathbf{i}T = \varnothing = \mathbf{o}S \cap \mathbf{o}T$

(ii) $\mathbf{i}S \setminus \mathbf{o}T = \mathbf{i}S \setminus \mathbf{a}T$ and $\mathbf{o}S \setminus \mathbf{i}T = \mathbf{o}S \setminus \mathbf{a}T$

(iii) $\mathbf{o}(S\ \mathbf{w}\ T) = \mathbf{o}(S\ \mathbf{b}\ T) = (\mathbf{o}S \setminus \mathbf{i}T) \cup (\mathbf{o}T \setminus \mathbf{i}S)$ and
$\mathbf{i}(S\ \mathbf{w}\ T) = \mathbf{i}(S\ \mathbf{b}\ T) = (\mathbf{i}S \setminus \mathbf{o}T) \cup (\mathbf{i}T \setminus \mathbf{o}S)$

(iv) $\mathbf{o}(S\ \mathbf{w}\ T) \cup \mathbf{i}(S\ \mathbf{w}\ T) = \mathbf{o}(S\ \mathbf{b}\ T) \cup \mathbf{i}(S\ \mathbf{b}\ T) = \mathbf{a}S \div \mathbf{a}T = \mathbf{a}(S\ \mathbf{b}\ T)$

(End of Property)

Lemma 3.5 : For connectable $\mathbf{C}_4$'s S and T, for trace $s \in \mathbf{t}(S\ \mathbf{w}\ T)$, and for symbol $a \in \mathbf{i}(S\ \mathbf{w}\ T)$

$$sa\lceil(\mathbf{a}S \div \mathbf{a}T) \in \mathbf{t}(S\ \mathbf{w}\ T)\lceil(\mathbf{a}S \div \mathbf{a}T) = sa \in \mathbf{t}(S\ \mathbf{w}\ T)$$

Proof : Without loss of generality we assume $a \in \mathbf{a}S$ and, hence, according to Property 3.0 (i), (ii), and (iii)

$$a \in \mathbf{i}S \setminus \mathbf{a}T \qquad (0)$$

$sa \in \mathbf{t}(S\ \mathbf{w}\ T)$

$\Rightarrow$ { projection on $\mathbf{a}S \div \mathbf{a}T$ }

$sa\lceil(\mathbf{a}S \div \mathbf{a}T) \in \mathbf{t}(S \mathbf{w} T)\lceil(\mathbf{a}S \div \mathbf{a}T)$

$\Rightarrow$ { One of the premises is $s \in \mathbf{t}(S \mathbf{w} T)$. Property 1.8, using the definition of blending }

$s \in \mathbf{t}(S \mathbf{w} T) \wedge sa\lceil(\mathbf{a}S \div \mathbf{a}T)\lceil(\mathbf{a}S \setminus \mathbf{a}T) \in \mathbf{t}S\lceil(\mathbf{a}S \setminus \mathbf{a}T)$

$\Rightarrow$ { definition of weaving and distribution of projection over concatenation, using Property 1.1 and $\mathbf{a}S \setminus \mathbf{a}T \subseteq \mathbf{a}S \div \mathbf{a}T$ and $\mathbf{a}S \setminus \mathbf{a}T \subseteq \mathbf{a}S$ }

$s\lceil\mathbf{a}S \in \mathbf{t}S \wedge s\lceil\mathbf{a}T \in \mathbf{t}T \wedge (s\lceil\mathbf{a}S)a\lceil(\mathbf{a}S \setminus \mathbf{a}T) \in \mathbf{t}S\lceil(\mathbf{a}S \setminus \mathbf{a}T)$

$\Rightarrow$ { $\mathbf{a}S \setminus \mathbf{a}T$ is independent with respect to S, since S and T are connectable, and $a \in (\mathbf{a}S \setminus \mathbf{a}T) \cap \mathbf{i}S$ according to (0) }

$(s\lceil\mathbf{a}S)a \in \mathbf{t}S \wedge s\lceil\mathbf{a}T \in \mathbf{t}T$

$=$ { distribution of projection over concatenation, using (0) }

$sa\lceil\mathbf{a}S \in \mathbf{t}S \wedge sa\lceil\mathbf{a}T \in \mathbf{t}T$

$=$ { definition of weaving, using $s \in (\mathbf{a}S \cup \mathbf{a}T)^*$ and $a \in \mathbf{a}S \cup \mathbf{a}T$ }

$sa \in \mathbf{t}(S \mathbf{w} T)$

(End of Proof)

4

Internal communications and external specification

The main issue of this chapter is to show absence of transmission and computation interference under composition of connectable C_4's. Since interference is a physical notion for mechanisms that send and receive signals, we begin this chapter with the introduction of a mechanistic appreciation of composition. In the last section it is argued that the blend is an operator for the specification of the composite that is in accordance with this mechanistic appreciation.

4.0. An informal mechanistic appreciation

We consider a mechanism and its environment that communicate with one another by sending and receiving signals. There are two types of signals : from the environment to the mechanism, the so-called inputs, and from the mechanism to the environment, which we call outputs. We assume that signals are conveyed via (finitely many) wires. With each wire we associate a symbol. A signal via a wire is denoted by its associated symbol.

A trace structure is viewed as the specification of such a mechanism-environment pair. Each symbol of the trace structure's alphabet corresponds to one wire. The alphabet is partitioned into an input and an output alphabet.

A trace is conceived as a sequence of events. Due to the concurrency of signals and the dependency of observations upon the position, there might not exist a unique sequence of events that describes the history of a mechanism-environment pair in operation. This history is rather described, at any time during operation, by a set, or equivalence class, of sequences of events. Traces that differ from one another because of the concurrency of symbols belong to the same equivalence class. Yet we associate, at any time during operation, one single trace, being a sequence of events, with the operation of a mechanism-environment pair. This trace is called the trace thus far generated. Since the discussion in the sequel relates to an arbitrary trace thus far generated it pertains, in fact, to the equivalence class of sequences of events.

Initially, the trace thus far generated is ϵ. The operation of the mechanism-environment pair corresponds to the generation of symbols. Each signal that the mechanism and environment communicate with one another can be viewed as the extension of the trace thus far generated with the symbol that is associated with that signal. Notice that only those extensions are allowed that yield a trace that belongs to the trace structure again.

We say that output symbols in the trace thus far generated have been sent and input symbols have been received by the mechanism. Whenever more convenient, we say that these symbols have been sent or received by the trace thus far generated instead of by the mechanism.

Under composition of two mechanism, wires to which the same symbol corresponds are connected. A wire that conveys input signals to the one mechanism should convey output signals from the other one. Accordingly, under composition of two trace structures, a common symbol is an input symbol of the one and an output symbol of the other trace structure. Composition can be viewed as replacing (a part of) one mechanism's environment by the other mechanism-environment pair.

We assume the so-called causality rule for mechanisms, i.e. no input signal can be received before the corresponding output signal has been sent. For the mechanistic appreciation of composition this means the following. At any instant, there are two traces thus far generated, one for each of the mechanism-environment pairs. Each trace may be extended with a symbol in the way described above, under the additional restriction that for each common symbol the number of times it has been received by the one trace does not exceed the number of times it has been sent by the other one.

A symbol sent by the one trace that has not been received by the other one is said to be on its way. Any two traces that can be brought about observing the restrictions above are called composable.

Absence of transmission and computation interference can be expressed in terms of composable traces. There is absence of transmission interference if we have for all pairs of composable traces : the number of occurrences of a common symbol sent by the one trace exceeds the number of occurrences of that symbol received by the other trace by at most one. There is absence of computation interference if we have for all pairs of composable traces : a symbol on its way from one trace to the other can be received by the latter, i.e. the extension of the latter trace with this symbol belongs to the trace structure of the corresponding mechanism.

We prove in the next sections that there is absence of computation and transmission interference under composition of two connectable C_4's. Therefore, we believe that the formal properties of delay-insensitivity and connectability provide a model that can be usefully applied to the problem of composing physical circuits and deriving the specification for the resulting circuit from the specifications of the composing circuits.

4.1. Formalization of the mechanistic appreciation

In this section we formalize the mechanistic appreciation as introduced above and the proof obligations for showing absence of transmission and computation interference.

Definition 4.0 : For connectable trace structures T and U, the composability of traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$, denoted by $\mathbf{c}(t,u)$, is defined by

$$\begin{aligned} & t = \epsilon \wedge u = \epsilon \ \vee \\ & (\exists a, t_0 :: t = t_0 a \ \wedge \ \mathbf{c}(t_0, u) \ \wedge \ (a \in \mathbf{o}\,U \Rightarrow \#_a u > \#_a t_0)) \ \vee \\ & (\exists b, u_0 :: u = u_0 b \ \wedge \ \mathbf{c}(t, u_0) \ \wedge \ (b \in \mathbf{o}\,T \Rightarrow \#_b t > \#_b u_0)) \end{aligned}$$

(End of Definition)

Notice that $\mathbf{c}(t,u) = \mathbf{c}(u,t)$. Notice also that $a \in \mathbf{o}\,U$ and $t = t_0 a$ implies $a \in \mathbf{a}\,T \cap \mathbf{a}\,U$ and, hence, on account of the definition of connectability, $a \in \mathbf{i}\,T$.

To cope with the various appearances of the arguments of $\mathbf{c}$ we state the following properties, which can readily be derived from the definition of $\mathbf{c}$. When referring to the definition of $\mathbf{c}$, one of the following properties may be meant.

Property 4.0 :

(i) $$\begin{aligned} \mathbf{c}(ta, ub) = {} & \mathbf{c}(t, ub) \wedge (a \in \mathbf{o}\,U \Rightarrow \#_a ub > \#_a t) \ \vee \\ & \mathbf{c}(ta, u) \wedge (b \in \mathbf{o}\,T \Rightarrow \#_b ta > \#_b u) \end{aligned}$$

(ii) $$\begin{aligned} \mathbf{c}(ta, u) = {} & \mathbf{c}(t, u) \wedge (a \in \mathbf{o}\,U \Rightarrow \#_a u > \#_a t) \ \vee \\ & (\exists b, u_0 :: u = u_0 b \ \wedge \ \mathbf{c}(ta, u_0) \ \wedge \\ & \qquad (b \in \mathbf{o}\,T \Rightarrow \#_b ta > \#_b u_0)) \end{aligned}$$

(iii) $\mathbf{c}(t,\epsilon) = (t \lceil \mathbf{o}\,U = \epsilon)$

(End of Property)

The two theorems that we have to prove are

Theorem 4.0 : (Absence of transmission interference) For connectable $\mathbf{C}_4$'s T and U, for composable traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$, and for symbol $a \in \mathbf{o}\,T \cap \mathbf{i}\,U$
$\#_a t - \#_a u \leq 1$

Theorem 4.1 : (Absence of computation interference) For connectable $\mathbf{C}_4$'s T and U, for composable traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$, and for symbol $a \in \mathbf{o}\,T \cap \mathbf{i}\,U$ such that $\#_a t > \#_a u$: $ua \in \mathbf{t}\,U$.

These two theorems are proved in the next section. We conclude this section with a few lemmata on composable traces.

Lemma 4.0 : For connectable $\mathbf{C}_4$'s T and U, for traces t and u, and for symbol a such that $ta \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$

$$\mathbf{c}(ta,u) \wedge a \notin \mathbf{i}\,U \Rightarrow \mathbf{c}(t,u)$$

Proof : By mathematical induction on the length of u.

Base : $u = \epsilon$.

$$
\begin{array}{ll}
 & \mathbf{c}(ta,u) \wedge a \notin \mathbf{i}\,U \\
\Rightarrow & \{ u = \epsilon \} \\
 & \mathbf{c}(ta,\epsilon) \\
= & \{ \text{definition of } \mathbf{c} \} \\
 & \mathbf{c}(t,\epsilon) \\
= & \{ u = \epsilon \} \\
 & \mathbf{c}(t,u)
\end{array}
$$

Step : $u = u_0 b$. Now we derive

$$
\begin{array}{ll}
 & \mathbf{c}(ta,u) \wedge a \notin \mathbf{i}\,U \\
= & \{ u = u_0 b \} \\
 & \mathbf{c}(ta,u_0 b) \wedge a \notin \mathbf{i}\,U \\
= & \{ \text{definition of } \mathbf{c} \text{ and calculus} \} \\
 & \mathbf{c}(t,u_0 b) \wedge (a \in \mathbf{o}\,U \Rightarrow \#_a u_0 b > \#_a t) \wedge a \notin \mathbf{i}\,U \ \vee \\
 & \mathbf{c}(ta,u_0) \wedge (b \in \mathbf{o}\,T \Rightarrow \#_b ta > \#_b u_0) \wedge a \notin \mathbf{i}\,U \\
\Rightarrow & \{ \text{calculus, using } u = u_0 b \text{, and the induction hypothesis} \} \\
 & \mathbf{c}(t,u) \vee \mathbf{c}(t,u_0) \wedge (b \in \mathbf{o}\,T \Rightarrow \#_b ta > \#_b u_0) \wedge a \notin \mathbf{i}\,U
\end{array}
$$

$\Rightarrow$ { calculus, using that $b \in \mathbf{a}\, U$ and $b \in \mathbf{o}\, T$ implies, by the connectability of T and U, $b \in \mathbf{i}\, U$ }

$\quad \mathbf{c}(t,u) \vee \mathbf{c}(t,u_0) \wedge b \notin \mathbf{o}\, T \vee \mathbf{c}(t,u_0) \wedge \#_b ta > \#_b u_0 \wedge a \neq b$

$\Rightarrow$ { definition of $\mathbf{c}$, using $u = u_0 b$, and calculus }

$\quad \mathbf{c}(t,u) \vee \mathbf{c}(t,u_0) \wedge \#_b t > \#_b u_0$

$\Rightarrow$ { definition of $\mathbf{c}$, using $u = u_0 b$ }

$\quad \mathbf{c}(t,u)$

(End of Proof)

Lemma 4.1 : For connectable C_4's T and U, for traces t and u, and for symbol a such that $ta \in \mathbf{t}\, T$ and $u \in \mathbf{t}\, U$

$$\mathbf{c}(ta,u) \wedge \#_a ta > \#_a u \Rightarrow \mathbf{c}(t,u)$$

Proof : By mathematical induction on the length of u.

Base : $u = \epsilon$.

$\quad \mathbf{c}(ta,u) \wedge \#_a ta > \#_a u$

$=$ { $u = \epsilon$ }

$\quad \mathbf{c}(ta,\epsilon)$

$\Rightarrow$ { definition of $\mathbf{c}$ }

$\quad \mathbf{c}(t,\epsilon)$

$=$ { $u = \epsilon$ }

$\quad \mathbf{c}(t,u)$

Step : $u = u_0 b$. Now we derive

$\quad \mathbf{c}(ta,u) \wedge \#_a ta > \#_a u$

$=$ { $u = u_0 b$ }

$\quad \mathbf{c}(ta,u_0 b) \wedge \#_a ta > \#_a u_0 b$

$=$ { definition of $\mathbf{c}$ and calculus }

$\quad \mathbf{c}(t,u_0 b) \wedge (a \in \mathbf{o}\, U \Rightarrow \#_a u_0 b > \#_a t) \wedge \#_a ta > \#_a u_0 b \vee$

$\quad \mathbf{c}(ta,u_0) \wedge (b \in \mathbf{o}\, T \Rightarrow \#_b ta > \#_b u_0) \wedge \#_a ta > \#_a u_0 b$

$\Rightarrow$ { calculus, using $u = u_0 b$ }

$\quad \mathbf{c}(t,u) \vee$

$\quad \mathbf{c}(ta,u_0) \wedge (b \in \mathbf{o}\, T \Rightarrow \#_b ta > \#_b u_0) \wedge \#_a ta > \#_a u_0 b \wedge a = b \vee$

$\quad \mathbf{c}(ta,u_0) \wedge (b \in \mathbf{o}\, T \Rightarrow \#_b ta > \#_b u_0) \wedge \#_a ta > \#_a u_0 b \wedge a \neq b$

$\Rightarrow$ { calculus }

$\quad \mathbf{c}(t,u) \vee \mathbf{c}(ta,u_0) \wedge \#_a t > \#_a u_0 \wedge a = b \vee$

$\quad \mathbf{c}(ta,u_0) \wedge (b \in \mathbf{o}T \Rightarrow \#_b t > \#_b u_0) \wedge \#_a ta > \#_a u_0$

$\Rightarrow$ { induction hypothesis and calculus }

$\quad \mathbf{c}(t,u) \vee \mathbf{c}(t,u_0) \wedge \#_b t > \#_b u_0 \vee \mathbf{c}(t,u_0) \wedge (b \in \mathbf{o}T \Rightarrow \#_b t > \#_b u_0)$

$\Rightarrow$ { definition of $\mathbf{c}$, using $u = u_0 b$ }

$\quad \mathbf{c}(t,u)$

(End of Proof)

Lemma 4.2 : For connectable $\mathbf{C}_4$'s T and U, for traces $t \in \mathbf{t}T$ and $u \in \mathbf{t}U$, and for symbol $a \in \mathbf{a}T \cap \mathbf{a}U$

$$\mathbf{c}(t,u) \wedge \#_a t > \#_a u \Rightarrow a \in \mathbf{o}T \cap \mathbf{i}U$$

Proof : By mathematical induction on $\mathbf{l}t + \mathbf{l}u$.

Base : $\mathbf{l}t + \mathbf{l}u = 0$. Then $\#_a t = \#_a u$.

Step : $\mathbf{l}t + \mathbf{l}u = k$, for some k, $k \geqslant 1$. Now we derive

$\quad \mathbf{c}(t,u) \wedge \#_a t > \#_a u$

$=$ { definition of $\mathbf{c}$, using $\neg(t = \epsilon \wedge u = \epsilon)$ since $k \geqslant 1$. Calculus }

$\quad (\exists b, t_0 :: t = t_0 b \wedge \mathbf{c}(t_0,u) \wedge (b \in \mathbf{o}U \Rightarrow \#_b u > \#_b t_0) \wedge \#_a t_0 b > \#_a u) \vee$

$\quad (\exists b, u_0 :: u = u_0 b \wedge \mathbf{c}(t,u_0) \wedge (b \in \mathbf{o}T \Rightarrow \#_b t > \#_b u_0) \wedge \#_a t > \#_a u_0 b)$

$\Rightarrow$ { Induction hypothesis applied to the second disjunct. Calculus }

$\quad (\exists b, t_0 :: b = a \wedge (b \in \mathbf{o}U \Rightarrow \#_b u > \#_b t_0) \wedge \#_a t_0 b > \#_a u) \vee$

$\quad (\exists b, t_0 :: b \neq a \wedge \mathbf{c}(t_0,u) \wedge \#_a t_0 b > \#_a u) \vee a \in \mathbf{o}T \cap \mathbf{i}U$

$\Rightarrow$ { Induction hypothesis applied to the second disjunct. Calculus }

$\quad (\exists t_0 :: (a \in \mathbf{o}U \Rightarrow \#_a u > \#_a t_0) \wedge \#_a t_0 \geqslant \#_a u) \vee a \in \mathbf{o}T \cap \mathbf{i}U$

$\Rightarrow$ { calculus, using $a \in \mathbf{a}T \cap \mathbf{a}U$ and the connectability of T and U on account of which $a \notin \mathbf{o}U = a \in \mathbf{o}T \cap \mathbf{i}U$ }

$\quad a \in \mathbf{o}T \cap \mathbf{i}U$

(End of Proof)

From Lemma 4.2 we infer the following corollary, using the symmetry of $\mathbf{c}$ and $(\mathbf{o}T \cap \mathbf{i}U) \cap (\mathbf{o}U \cap \mathbf{i}T) = \varnothing$.

Corollary 4.0 : For connectable C_4's T and U, for traces $t \in \mathbf{t}T$ and $u \in \mathbf{t}U$, and for symbol $a \in \mathbf{a}T \cap \mathbf{a}U$

$$\mathbf{c}(t,u) \wedge a \in \mathbf{o}T \cap \mathbf{i}U \Rightarrow \#_a t \geqslant \#_a u$$

4.2. Absence of transmission and computation interference

(This section may be skipped on first reading.) In this section we prove the absence of transmission and computation interference. To that end we consider, for connectable trace structures T and U and for composable traces $t \in \mathbf{t}T$ and $u \in \mathbf{t}U$, symbols on their way from one trace to the other. Rather than considering these symbols individually, we consider the set of sequences of symbols, called traces again, consisting of the symbols on their way in one direction.

Definition 4.1 : For connectable trace structures T and U, and for composable traces $t \in \mathbf{t}T$ and $u \in \mathbf{t}U$ we define $\mathbf{from}(t,u)$ as

$$\{x : x \in (\mathbf{o}T \cap \mathbf{i}U)^* \wedge (\forall a : a \in \mathbf{o}T \cap \mathbf{i}U : \#_a x = \#_a t - \#_a u) : x\}$$

(End of Definition)

Consequently, $\mathbf{from}(t,u)$ is the set of traces that are a permutation of all symbols sent by t and not received by u. Since $\#_a t \geqslant \#_a u$ for $a \in \mathbf{o}T \cap \mathbf{i}U$ on account of Corollary 4.0, $\mathbf{from}(t,u)$ is non-empty. Since the lengths of the traces in $\mathbf{from}(t,u)$ are equal, we define $\mathbf{l}(\mathbf{from}(t,u))$ as the length of the traces in $\mathbf{from}(t,u)$, which is the number of symbols on their way from t to u.

The total number of symbols on their way between t and u is called the number of mismatches and is denoted by $\mathbf{mm}(t,u)$.

Definition 4.2 : For connectable trace structures T and U, and for composable traces $t \in \mathbf{t}T$ and $u \in \mathbf{t}U$

$$\mathbf{mm}(t,u) = \mathbf{l}(\mathbf{from}(t,u)) + \mathbf{l}(\mathbf{from}(u,t))$$

(End of Definition)

We shall frequently use the following properties of **from** and **mm**. Proofs are omitted but can be derived using the definitions and Lemmata 4.0 and 4.2.

Property 4.1 : For connectable trace structures T and U, and for composable traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$

(i) $u_0 \in \mathbf{from}(t,u) \wedge uu_0 \in \mathbf{t}\,U \Rightarrow \mathbf{c}(t,uu_0)$

(ii) $u_0 \in \mathbf{from}(t,u) \wedge uu_0 \in \mathbf{t}\,U \Rightarrow \mathbf{from}(t,uu_0) = \{\epsilon\}$

(iii) $u = u_0u_1 \wedge u_1 \in \mathbf{o}\,T^* \Rightarrow \mathbf{mm}(t,u_0) = \mathbf{mm}(t,u_0u_1) + \mathbf{l}\,u_1$

(iv) $u = u_0u_1 \wedge u_1 \in \mathbf{o}\,T^* \wedge u_2 \in \mathbf{from}(t,u_0u_1) \Rightarrow u_1u_2 \in \mathbf{from}(t,u_0)$

(v) $u = u_0u_1 \wedge u_1 \lceil \mathbf{i}\,T = \epsilon \Rightarrow \mathbf{from}(u_0u_1,t) = \mathbf{from}(u_0,t)$

(End of Property)

In order to prove Theorems 4.0 and 4.1, the absence of transmission and computation interference respectively, we prove the following lemma.

Lemma 4.3 : For connectable trace structures T and U such that $\mathbf{a}\,T = \mathbf{a}\,U$, and for composable traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$

$$(\forall u_0 : u_0 \in \mathbf{from}(t,u) : uu_0 \in \mathbf{t}\,U)$$

Theorems 4.0 and 4.1 are derived from this lemma in the following way. Let T and U be connectable $\mathbf{C}_4$'s and let $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$ be composable traces. Since $\mathbf{a}\,T \cap \mathbf{a}\,U$ is independent with respect to both T and U, $T\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$ and $U\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$ are $\mathbf{C}_4$'s as well according to Lemma 3.4. Moreover, their alphabets are equal, viz. $\mathbf{a}\,T \cap \mathbf{a}\,U$, and they are connectable as follows from the definition of connectability.

From the definition of composability it can be seen that the strings of common symbols in t and u determine both the composability of t and u and the symbols on their way from t to u. Hence, from the composability of t and u with respect to T and U we infer the composability of $t\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$ and $u\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$ with respect to $T\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$ and $U\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$. Let u_0 be a string of all symbols on their way from t to u. Then u_0 is a string of all symbols on their way from $t\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$ to $u\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$ as well. This implies that $(u\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U))u_0 \in \mathbf{t}\,U\lceil(\mathbf{a}\,T \cap \mathbf{a}\,U)$ according to Lemma 4.3. The symbols of u_0 are input symbols to U and belong to $\mathbf{a}\,T \cap \mathbf{a}\,U$. Since $\mathbf{a}\,T \cap \mathbf{a}\,U$ is independent with respect to U and since $u \in \mathbf{t}\,U$ we conclude, by applying Definition 3.0 a number of times, that $uu_0 \in \mathbf{t}\,U$.

Notice that u_0 is an arbitrary permutation of the symbols on their way from T to U, i.e. of symbols $a \in \mathbf{o}\,T \cap \mathbf{i}\,U$ with $\#_a t > \#_a u$. First, $uu_0 \in \mathbf{t}\,U$ then implies that each symbol occurs at most once in u_0, since, on account of $\mathbf{R}_2$, adjacent symbols are distinct. Hence, we have proved absence of transmission interference. Second, again since u_0 is an arbitrary permutation of symbols on their way and, hence, may start with any symbol on its way from t to u, it implies absence of computation interference, since $\mathbf{t}\,U$ is prefix-closed.

Lemma 4.3 is proved by mathematical induction. In order to reduce the length of the proof we first prove two additional lemmata, in which we assume the induction hypothesis for Lemma 4.3. Let, for the remainder of this section T and U be connectable $\mathbf{C}_4$'s such that $\mathbf{a}\,T = \mathbf{a}\,U$. Consequently, $\mathbf{i}\,T = \mathbf{o}\,U$, $\mathbf{o}\,T = \mathbf{i}\,U$, and $\mathbf{t}\,T = \mathbf{t}\,U$.

Lemma 4.4 : Given integer k and given that all composable traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$ with $\mathbf{l}\,t + \mathbf{l}\,u + \mathbf{mm}(t,u) \leq k$ satisfy

$$(\forall u_0 : u_0 \in \mathbf{from}(t,u) : uu_0 \in \mathbf{t}\,U) \wedge (\forall t_0 : t_0 \in \mathbf{from}(u,t) : tt_0 \in \mathbf{t}\,T)$$

Then for traces s, t, u, v, and w, and for symbol $a \in \mathbf{o}\,T$ such that $t \in \mathbf{o}\,U^*$, $v \in \mathbf{o}\,T^*$, $satw \in \mathbf{t}\,T$, and $uavw \in \mathbf{t}\,U$

$$\begin{aligned} &\mathrm{c}(satw, uavw) \wedge \mathbf{l}(satw) + \mathbf{l}(uavw) + \mathbf{mm}(satw, uavw) \leq k \\ &\Rightarrow staw \in \mathbf{t}\,T \wedge uvaw \in \mathbf{t}\,U \wedge \mathrm{c}(staw, uvaw) \end{aligned}$$

Proof : By mathematical induction on the length of w.

Base : $w = \epsilon$. We assume the left-hand side of the implication, hence,

$$\mathrm{c}(sat, uav) \text{ and } \mathbf{l}(sat) + \mathbf{l}(uav) + \mathbf{mm}(sat, uav) \leq k \tag{0}$$

Let t_0 and u_0 be such that

$$t_0 \in \mathbf{from}(uav, sat) \text{ and } u_0 \in \mathbf{from}(sat, uav) \tag{1}$$

Since $\mathbf{t}\,T$ is prefix-closed, and since a and the symbols of v are of the same type, which makes $\mathbf{R}_3$ applicable, we have

$$sa \in \mathbf{t}\,T \text{ and } uva \in \mathbf{t}\,U \tag{2}$$

Now we derive

true

$=$ { (0) and (1) }

$\mathrm{c}(sat, uav) \wedge \mathbf{l}(sat) + \mathbf{l}(uav) + \mathbf{mm}(sat, uav) \leq k \wedge$

$t_0 \in \mathbf{from}(uav, sat) \ \wedge \ u_0 \in \mathbf{from}(sat, uav)$

$\Rightarrow$ { Lemma 4.0, Property 4.1 (iii), (iv), and (v), using $t \in \mathbf{o}\,U^*$ }

$\mathbf{c}(sa, uav) \ \wedge \ \mathbf{l}(sa) + \mathbf{l}(uav) + \mathbf{mm}(sa, uav) \leqslant k \ \wedge$

$tt_0 \in \mathbf{from}(uav, sa) \ \wedge \ u_0 \in \mathbf{from}(sa, uav)$

$\Rightarrow$ { Lemma 4.0, Corollary 4.0, and Property 4.1 (iii), (iv), and (v), using $av \in \mathbf{o}\,T^*$ }

$\mathbf{c}(sa, u) \ \wedge \ \#_a sa > \#_a u \ \wedge \ \mathbf{l}(sa) + \mathbf{l}u + \mathbf{mm}(sa, u) \leqslant k \ \wedge$

$tt_0 \in \mathbf{from}(u, sa) \ \wedge \ avu_0 \in \mathbf{from}(sa, u)$

$\Rightarrow$ { Lemma 4.1, definition of **mm** and **from**, using $a \in \mathbf{o}\,T$ }

$\mathbf{c}(s, u) \ \wedge \ \mathbf{l}s + \mathbf{l}u + \mathbf{mm}(s, u) \leqslant k - 2 \ \wedge \ tt_0 \in \mathbf{from}(u, s) \ \wedge \ vu_0 \in \mathbf{from}(s, u)$

$\Rightarrow$ { premise }

$stt_0 \in \mathbf{t}\,T \ \wedge \ uvu_0 \in \mathbf{t}\,U \ \wedge \ \mathbf{c}(s, u) \ \wedge \ tt_0 \in \mathbf{from}(u, s) \ \wedge \ vu_0 \in \mathbf{from}(s, u)$

$\Rightarrow$ { Property 4.1 (i) and (v), using $tt_0 \in \mathbf{o}\,U^*$ and $vu_0 \in \mathbf{o}\,T^*$ }

$stt_0 \in \mathbf{t}\,T \ \wedge \ uvu_0 \in \mathbf{t}\,U \ \wedge \ \mathbf{c}(stt_0, uvu_0)$

$\Rightarrow$ { $\mathbf{t}\,T$ and $\mathbf{t}\,U$ are prefix-closed, Lemma 4.0, using $t_0 \in \mathbf{o}\,U^*$ and $u_0 \in \mathbf{o}\,T^*$ }

$st \in \mathbf{t}\,T \ \wedge \ uv \in \mathbf{t}\,U \ \wedge \ \mathbf{c}(st, uv)$

$\Rightarrow$ { Lemma 2.2, using (2) and the fact that a and the symbols of t are of different types }

$sta \in \mathbf{t}\,T \ \wedge \ uva \in \mathbf{t}\,U \ \wedge \ \mathbf{c}(st, uv)$

$\Rightarrow$ { definition of $\mathbf{c}$, using $\#_a st \geqslant \#_a uv$ according to Corollary 4.0 }

$sta \in \mathbf{t}\,T \ \wedge \ uva \in \mathbf{t}\,U \ \wedge \ \mathbf{c}(sta, uva)$

$=$ { $w = \epsilon$ }

$staw \in \mathbf{t}\,T \ \wedge \ uvaw \in \mathbf{t}\,U \ \wedge \ \mathbf{c}(staw, uvaw)$

Step : $w = w_0c$. Assuming the left-hand side of the implication again, we have

$$\begin{aligned} &\mathbf{c}(satw_0c, uavw_0c) \ \text{ and} \\ &\mathbf{l}(satw_0c) + \mathbf{l}(uavw_0c) + \mathbf{mm}(satw_0c, uavw_0c) \leqslant k \end{aligned} \tag{3}$$

Now we derive

true

$=$ { (3) }

$satw_0c \in \mathbf{t}\,T \ \wedge \ uavw_0c \in \mathbf{t}\,U \ \wedge \ \mathbf{c}(satw_0c, uavw_0c) \ \wedge$

$\mathbf{l}(satw_0c) + \mathbf{l}(uavw_0c) + \mathbf{mm}(satw_0c, uavw_0c) \leqslant k$

$\Rightarrow$ { Lemmata 4.0 and 4.1, Corollary 4.0, and the definition of **mm** }

$satw_0c \in \mathbf{t}\,T \ \wedge \ uavw_0c \in \mathbf{t}\,U \ \wedge \ \mathbf{c}(satw_0, uavw_0) \ \wedge$

$\mathbf{l}(satw_0)+\mathbf{l}(uavw_0)+\mathbf{mm}(satw_0,uavw_0)\leqslant k-2$

$\Rightarrow$ { induction hypothesis and the definition of $\mathbf{mm}$. Moreover, a and the symbols of v are of the same type, which makes $\mathbf{R}_3$ applicable }

$$satw_0c\in\mathbf{t}\,T\ \wedge\ staw_0\in\mathbf{t}\,T\ \wedge\ uwaw_0c\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw_0,uwaw_0)\ \wedge$$
$$\mathbf{l}(staw_0)+\mathbf{l}(uwaw_0)+\mathbf{mm}(staw_0,uwaw_0)\leqslant k-2 \tag{4}$$

We distinguish two cases : (i) $c\in\mathbf{o}\,T$ and (ii) $c\in\mathbf{o}\,U$

(i) $c\in\mathbf{o}\,T$

true

$=$ { (4), applying Lemma 2.6, using that the symbols of t are of another type than a and c; calculus }

$staw_0c\in\mathbf{t}\,T\ \wedge\ uwaw_0c\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw_0,uwaw_0)$

$\Rightarrow$ { definition of $\mathbf{c}$, using $\#_c staw_0\geqslant\#_c uwaw_0$ according to Corollary 4.0 }

$staw_0c\in\mathbf{t}\,T\ \wedge\ uwaw_0c\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw_0c,uwaw_0c)$

$=$ { $w_0c\ =\ w$ }

$staw\in\mathbf{t}\,T\ \wedge\ uwaw\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw,uwaw)$

(ii) $c\in\mathbf{o}\,U$

true

$=$ { (4) and calculus }

$staw_0\in\mathbf{t}\,T\ \wedge\ uwaw_0c\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw_0,uwaw_0)\ \wedge$

$\mathbf{l}(staw_0)+\mathbf{l}(uwaw_0)+\mathbf{mm}(staw_0,uwaw_0)\leqslant k-2$

$\Rightarrow$ { definition of $\mathbf{c}$ and $\mathbf{mm}$, using $c\in\mathbf{o}\,U$ and $\#_c uwaw_0\geqslant\#_c staw_0$ according to Corollary 4.0 }

$staw_0\in\mathbf{t}\,T\ \wedge\ uwaw_0c\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw_0,uwaw_0c)\ \wedge\ \#_c uwaw_0c>\#_c staw_0\ \wedge$

$\mathbf{l}(staw_0)+\mathbf{l}(uwaw_0c)+\mathbf{mm}(staw_0,uwaw_0c)\leqslant k$

$\Rightarrow$ { premise, using that $\mathbf{t}\,T$ is prefix-closed and that there is a trace in $\mathbf{from}(uwaw_0c,staw_0)$ that begins with c }

$staw_0c\in\mathbf{t}\,T\ \wedge\ uwaw_0c\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw_0,uwaw_0c)\ \wedge\ \#_c uwaw_0c>\#_c staw_0$

$\Rightarrow$ { definition of $\mathbf{c}$, using $c\in\mathbf{o}\,U$ }

$staw_0c\in\mathbf{t}\,T\ \wedge\ uwaw_0c\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw_0c,uwaw_0c)$

$=$ { $w\ =\ w_0c$ }

$staw\in\mathbf{t}\,T\ \wedge\ uwaw\in\mathbf{t}\,U\ \wedge\ \mathbf{c}(staw,uwaw)$

(End of Proof)

Lemma 4.5 : Given integer k and given that all composable traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$ with $\mathbf{l}t + \mathbf{l}u + \mathbf{mm}(t,u) \leqslant k$ satisfy

$$(\forall u_0 : u_0 \in \mathbf{from}(t,u) : uu_0 \in \mathbf{t}\,U) \wedge (\forall t_0 : t_0 \in \mathbf{from}(u,t) : tt_0 \in \mathbf{t}\,T)$$

Then for composable traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$ with $\mathbf{l}t + \mathbf{l}u = k-1$ and $\mathbf{mm}(t,u) = 0$

$$(\forall c : c \in \mathbf{o}\,T : tc \in \mathbf{t}\,T \Rightarrow uc \in \mathbf{t}\,U) \wedge (\forall c : c \in \mathbf{o}\,U : uc \in \mathbf{t}\,U \Rightarrow tc \in \mathbf{t}\,T)$$

Proof : We observe, since $\mathbf{a}\,T = \mathbf{a}\,U$, that the lengths of composable traces $t \in \mathbf{t}\,T$ and $u \in \mathbf{t}\,U$ with $\mathbf{mm}(t,u) = 0$ are equal. Moreover, if these traces are non-empty, at least one of them contains a symbol that is an output symbol for the trace structure to which that trace belongs.

We prove the theorem by mathematical induction on the length of the longest common suffix of t and u.

Base : The length of the longest common suffix of t and u equals $\mathbf{l}t$ and $\mathbf{l}u$. Then $t = u$ and the lemma holds, since $\mathbf{t}\,T = \mathbf{t}\,U$, due to the connectability of T and U.

Step :

$$t = t_0 w \ \text{ and } \ u = u_0 w \ \text{ with } \ \mathrm{c}(t,u),\ \mathbf{l}t + \mathbf{l}u = k-1, \quad \text{and} \ \ \mathbf{mm}(t,u) = 0 \tag{0}$$

Traces t_0 and u_0 are non-empty and do not end in the same symbol. Notice that we assume the lemma to hold for composable traces with a longest common suffix that is longer than w. Applying Lemmata 4.0 and 4.1, using the definition of **from**, we derive from (0) and the composability of t and u

$$\mathrm{c}(t_0,u_0) \ \text{ and } \ \mathbf{mm}(t_0,u_0) = 0 \tag{1}$$

We define r, t_2, s, and v in the following way.

$$t_0 = rt_2 \text{ and } u_0 = sv \text{ with } t_2 \in \mathbf{o}\,U^* \text{ and } v \in \mathbf{o}\,T^* \tag{2}$$

$$r \text{ and } v \text{ do not end in a symbol of } \mathbf{o}\,U \text{ and } \mathbf{o}\,T \text{ respectively} \tag{3}$$

Moreover we have, according to the observation made at the beginning of this proof

$$\mathbf{l}r + \mathbf{l}s > 0 \tag{4}$$

Now we derive

$$
\begin{array}{ll}
 & \text{true} \\
= & \{\ (1) \text{ and } (2)\ \} \\
 & \mathrm{c}(rt_2, sv) \wedge t_2 \in \mathbf{o}U^* \wedge v \in \mathbf{o}T^* \\
\Rightarrow & \{\ \text{Lemma 4.0}\ \} \\
 & \mathrm{c}(r,s) \qquad (5) \\
\Rightarrow & \{\ \text{definition of } \mathrm{c}\text{, using } (4)\ \} \\
 & (\exists t_1, a :: r = t_1 a \wedge \mathrm{c}(t_1, s) \wedge (a \in \mathbf{o}U \Rightarrow \#_a s > \#_a t_1)) \vee \\
 & (\exists u_1, a :: s = u_1 a \wedge \mathrm{c}(r, u_1) \wedge (a \in \mathbf{o}T \Rightarrow \#_a r > \#_a u_1))
\end{array}
$$

Without loss of generality we may, due to the symmetric formulation of this lemma, assume the first disjunct to hold and, hence, t_1 and a to be defined. Hence, we have

$$r = t_1 a \qquad (6)$$

$$\mathrm{c}(t_1, s) \qquad (7)$$

$$a \in \mathbf{o}T \qquad (8)$$

$$\mathrm{c}(t_1 a, s) \qquad (9)$$

(8) follows from (3) and (6), and (9) follows from (5) and (6). According to Corollary 4.0 we infer from (7) and (8) that $\#_a t_1 a > \#_a s$. Hence, from (6), (1), and (2) we conclude that v contains symbol a. Consequently, using (2), we may assume traces u_1 and u_2 to be defined such that

$$sv = u_1 a u_2 \text{ and } u_2 \in \mathbf{o}T^* \qquad (10)$$

Combining (0) through (10), we infer

$$t = t_1 a t_2 w \text{ and } u = u_1 a u_2 w \qquad (11)$$

$$a \in \mathbf{o}T \qquad (12)$$

$$t_2 \in \mathbf{o}U^* \text{ and } u_2 \in \mathbf{o}T^* \qquad (13)$$

$$\mathrm{c}(t_1 a t_2 w, u_1 a u_2 w) \qquad (14)$$

$$\mathrm{l}(t_1 a t_2 w) + \mathrm{l}(u_1 a u_2 w) = k-1 \text{ and } \mathrm{mm}(t_1 a t_2 w, u_1 a u_2 w) = 0 \qquad (15)$$

which allows us to apply Lemma 4.4 and derive

$$t_1 t_2 a w \in \mathbf{t}T \text{ and } u_1 u_2 a w \in \mathbf{t}U \text{ and } \mathrm{c}(t_1 t_2 a w, u_1 u_2 a w) \qquad (16)$$

Now we may apply the induction hypothesis, since the longest common suffix of t_1t_2aw and u_1u_2aw is at least aw, which is longer than w, and, moreover, $\mathbf{l}(t_1t_2aw)+\mathbf{l}(u_1u_2aw) = k-1$ and $\mathbf{mm}(t_1t_2aw, u_1u_2aw) = 0$, due to (15). Hence, we have

$$(\forall c : c \in \mathbf{o}\,T : t_1t_2awc \in \mathbf{t}\,T \Rightarrow u_1u_2awc \in \mathbf{t}\,U) \tag{17}$$

$$(\forall c : c \in \mathbf{o}\,U : u_1u_2awc \in \mathbf{t}\,U \Rightarrow t_1t_2awc \in \mathbf{t}\,T) \tag{18}$$

from which we derive for any $c \in \mathbf{o}\,T$

$$\begin{array}{ll}
 & tc \in \mathbf{t}\,T \\
= & \{ \text{ (11) and (16) } \} \\
 & t_1at_2wc \in \mathbf{t}\,T \ \wedge\ t_1t_2aw \in \mathbf{t}\,T \\
\Rightarrow & \{ \text{ Lemma 2.6, since } t_2 \in \mathbf{o}\,U^* \text{ according to (13), } a \in \mathbf{o}\,T \text{ according to (12),} \\
 & \quad \text{and } c \in \mathbf{o}\,T \ \} \\
 & t_1t_2awc \in \mathbf{t}\,T \\
\Rightarrow & \{ \text{ (17) } \} \\
 & u_1u_2awc \in \mathbf{t}\,U \\
= & \{ \ \mathbf{R}_3, \text{ since } a \in \mathbf{o}\,T \text{ and } u_2 \in \mathbf{o}\,T^* \text{ according to (12) and (13) } \} \\
 & u_1au_2wc \in \mathbf{t}\,U \\
= & \{ \text{ (11) } \} \\
 & uc \in \mathbf{t}\,U
\end{array}$$

and for any $c \in \mathbf{o}\,U$

$$\begin{array}{ll}
 & uc \in \mathbf{t}\,U \\
= & \{ \text{ (11) } \} \\
 & u_1au_2wc \in \mathbf{t}\,U \\
= & \{ \ \mathbf{R}_3, \text{ since } a \in \mathbf{o}\,T \text{ and } u_2 \in \mathbf{o}\,T^* \text{ according to (12) and (13) } \} \\
 & u_1u_2awc \in \mathbf{t}\,U \\
\Rightarrow & \{ \text{ (18) } \} \\
 & t_1t_2awc \in \mathbf{t}\,T \\
= & \{ \text{ (11) and } t \in \mathbf{t}\,T \ \} \\
 & t_1t_2awc \in \mathbf{t}\,T \ \wedge\ t_1at_2w \in \mathbf{t}\,T \\
\Rightarrow & \{ \text{ Lemma 2.7, since } t_2 \in \mathbf{o}\,U^* \text{ according to (13), } a \in \mathbf{o}\,T \text{ according to (12),} \\
 & \quad \text{and } c \in \mathbf{o}\,U \ \} \\
 & t_1at_2wc \in \mathbf{t}\,T
\end{array}$$

$= \{ (11) \}$
$u \in \mathbf{t}\, T$

(End of Proof)

Proof of Lemma 4.3 : We prove for composable traces $t \in \mathbf{t}\, T$ and $u \in \mathbf{t}\, U$

$$(\forall u_0 : u_0 \in \mathbf{from}(t,u) : uu_0 \in \mathbf{t}\, U) \wedge (\forall t_0 : t_0 \in \mathbf{from}(u,t) : tt_0 \in \mathbf{t}\, T)$$

by mathematical induction on $\mathbf{l}t + \mathbf{l}u + \mathbf{mm}(t,u)$.

Base : $\mathbf{l}t + \mathbf{l}u + \mathbf{mm}(t,u) \leqslant 1$. Obvious, since for composable traces $\mathbf{l}t + \mathbf{l}u + \mathbf{mm}(t,u) \neq 1$, and since $\epsilon \in \mathbf{t}\, T$ and $\epsilon \in \mathbf{t}\, U$.

Step : We assume, given an integer k, $k \geqslant 1$, that for composable traces $t \in \mathbf{t}\, T$ and $u \in \mathbf{t}\, U$ with $\mathbf{l}t + \mathbf{l}u + \mathbf{mm}(t,u) \leqslant k$

$$(\forall u_0 : u_0 \in \mathbf{from}(t,u) : uu_0 \in \mathbf{t}\, U) \wedge (\forall t_0 : t_0 \in \mathbf{from}(u,t) : tt_0 \in \mathbf{t}\, T) \quad (0)$$

Let $t \in \mathbf{t}\, T$ and $u \in \mathbf{t}\, U$ be composable traces such that

$$\mathbf{l}t + \mathbf{l}u + \mathbf{mm}(t,u) = k+1 \quad (1)$$

As in the proof of Lemma 4.5, we may assume, due to the symmetric formulation of this lemma

$$t = t_0 a t_1 \text{ and } u = u_0 u_1 \quad (2)$$

$$\mathbf{c}(t_0 a t_1, u_0 u_1) \quad (3)$$

$$a \in \mathbf{o}\, T \quad (4)$$

$$t_1 \in \mathbf{o}\, U^* \text{ and } u_1 \in \mathbf{o}\, T^* \quad (5)$$

$$\mathbf{c}(t_0, u_0) \quad (6)$$

From (1), (2), (3), and (5) we infer, using Lemma 4.0 and Property 4.1 (iii)

$$\mathbf{l}(t_0 a) + \mathbf{l}u_0 + \mathbf{mm}(t_0 a, u_0) = k+1 \quad (7)$$

From (4), (6), and (7) we infer

$$\mathbf{l}(t_0) + \mathbf{l}u_0 + \mathbf{mm}(t_0, u_0) = k-1 \quad (8)$$

We have to prove for traces $t_2 \in \mathbf{from}(u,t)$ and $u_2 \in \mathbf{from}(t,u)$ that $tt_2 \in \mathbf{t}\, T$ and $uu_2 \in \mathbf{t}\, U$. Let

$$t_2 \in \mathbf{from}(u,t) \text{ and } u_2 \in \mathbf{from}(t,u) \quad (9)$$

Now we derive

$$
\begin{array}{ll}
 & \text{true} \\
= & \{\ (2) \text{ and } (9)\ \} \\
 & t_2 \in \mathbf{from}(u_0u_1, t_0at_1) \\
\Rightarrow & \{\ \text{Property 4.1 (v), using } u_1 \in \mathbf{o}\,T^* \text{ according to (5)}\ \} \\
 & t_2 \in \mathbf{from}(u_0, t_0at_1) \\
\Rightarrow & \{\ \text{Property 4.1 (iv), using } t_1 \in \mathbf{o}\,U^* \text{ according to (5)}\ \} \\
 & t_1t_2 \in \mathbf{from}(u_0, t_0a) \qquad (10) \\
\Rightarrow & \{\ \text{definition of } \mathbf{from}, \text{ using } a \in \mathbf{o}\,T \text{ according to (4) and } \mathrm{c}(t_0, u_0) \text{ according} \\
 & \quad \text{to (6)}\ \} \\
 & t_1t_2 \in \mathbf{from}(u_0, t_0) \qquad (11) \\
\Rightarrow & \{\ \text{induction hypothesis, using (6) and (8)}\ \} \\
 & t_0t_1t_2 \in \mathbf{t}\,T \qquad (12) \\
= & \{\ (2), \text{ using the prefix-closedness of } \mathbf{t}\,T, \text{ and (4) and (11)}\ \} \\
 & t_0t_1t_2 \in \mathbf{t}\,T \ \wedge\ t_0a \in \mathbf{t}\,T \ \wedge\ a \in \mathbf{o}\,T \ \wedge\ t_1t_2 \in \mathbf{o}\,U^* \\
\Rightarrow & \{\ \text{Lemma 2.2}\ \} \\
 & t_0at_1t_2 \in \mathbf{t}\,T \ \wedge\ t_0t_1t_2a \in \mathbf{t}\,T \\
= & \{\ (2)\ \} \\
 & tt_2 \in \mathbf{t}\,T \ \wedge\ t_0t_1t_2a \in \mathbf{t}\,T \qquad (13)
\end{array}
$$

This means that we have proved half of the lemma, viz. $tt_2 \in \mathbf{t}\,T$.

In the same way as we derived $t_1t_2 \in \mathbf{from}(u_0, t_0a)$ (cf. (10)), we can also derive

$$u_1u_2 \in \mathbf{from}(t_0a, u_0) \tag{14}$$

The traces of $\mathbf{from}(t_0, u_0)$ contain one symbol a less than the traces of $\mathbf{from}(t_0a, u_0)$, since $a \in \mathbf{o}\,T$ according to (4). Let

$$u_3 \in \mathbf{from}(t_0, u_0) \tag{15}$$

Then we have that $u_3a \in \mathbf{from}(t_0a, u_0)$ and, hence, according to (14) and the definition of **from**, that u_1u_2 is a permutation of u_3a. We have to prove $uu_2 \in \mathbf{t}\,U$ or, equivalently by (2), $u_0u_1u_2 \in \mathbf{t}\,U$. By $\mathbf{R}_3$ it now suffices to prove $u_0u_3a \in \mathbf{t}\,U$, since all symbols of u_3a are of the same type. We derive

$$
\begin{array}{ll}
 & \text{true} \\
= & \{\ (15)\ \} \\
 & u_3 \in \mathbf{from}(t_0, u_0)
\end{array}
$$

$=$ { induction hypothesis, using (6) and (8) }

$u_3 \in \mathbf{from}(t_0, u_0) \wedge u_0 u_3 \in \mathbf{t}\, U$

$=$ { (11) and (12) }

$u_3 \in \mathbf{from}(t_0, u_0) \wedge u_0 u_3 \in \mathbf{t}\, U \wedge t_1 t_2 \in \mathbf{from}(u_0, t_0) \wedge t_0 t_1 t_2 \in \mathbf{t}\, T$

$\Rightarrow$ { Property 4.1 (i), (ii), and (v), using the definition of **from** }

$u_0 u_3 \in \mathbf{t}\, U \wedge t_1 t_2 \in \mathbf{from}(u_0 u_3, t_0) \wedge t_0 t_1 t_2 \in \mathbf{t}\, T \wedge$
$\mathbf{from}(t_0, u_0 u_3) = \{\epsilon\} \wedge u_3 \in \mathbf{o}\, T^* \wedge \mathbf{c}(t_0, u_0 u_3)$

$\Rightarrow$ { Property 4.1 (i), (ii), and (v), using the definitions of **mm** and **from** }

$t_0 t_1 t_2 \in \mathbf{t}\, T \wedge u_0 u_3 \in \mathbf{t}\, U \wedge \mathbf{c}(t_0 t_1 t_2, u_0 u_3) \wedge \mathbf{mm}(t_0 t_1 t_2, u_0 u_3) = 0 \wedge$
$u_3 \in \mathbf{o}\, T^* \wedge t_1 t_2 \in \mathbf{o}\, U^*$

$\Rightarrow$ { Lemma 4.5, using $\mathbf{l}(t_0 t_1 t_2) + \mathbf{l}(u_0 u_3) = k - 1$, which we derive from (8) and Property 4.1 (iii) }

$(\forall c : c \in \mathbf{o}\, T : t_0 t_1 t_2 c \in \mathbf{t}\, T \Rightarrow u_0 u_3 c \in \mathbf{t}\, U)$

$\Rightarrow$ { instantiation, using $a \in \mathbf{o}\, T$ according to (4), and $t_0 t_1 t_2 a \in \mathbf{t}\, T$ according to (13) }

$u_0 u_3 a \in \mathbf{t}\, U$

(End of Proof)

4.3. Blending as a composition operator

In the previous section we have proved the absence of transmission and computation interference. In this section we argue that blending as a composition operator is a proper abstraction of the mechanistic appreciation of composition as discussed earlier. We consider this a sufficient justification for using blending as a composition operator for composing C_4's by means of independent alphabets.

In the remainder of this section T and U are connectable C_4's. We define for two composable traces the set of resulting traces in the following way.

Definition 4.3 : For traces $t \in \mathbf{t}\, T$, $u \in \mathbf{t}\, U$, and x we say that x is a resultant of t and u, denoted by $x\, \mathbf{r}\, (t, u)$, if

$$x = \epsilon \wedge t = \epsilon \wedge u = \epsilon \vee$$
$$(\exists a, x_0, t_0 :: x = x_0 a \wedge t = t_0 a \wedge x_0 \mathbf{r}(t_0, u) \wedge (a \in \mathbf{o}\, U \Rightarrow \#_a u > \#_a t_0)) \vee$$
$$(\exists a, x_0, u_0 :: x = x_0 a \wedge u = u_0 a \wedge x_0 \mathbf{r}(t, u_0) \wedge (a \in \mathbf{o}\, T \Rightarrow \#_a t > \#_a u_0))$$

(End of Definition)

Composability of traces $t \in \mathbf{t}T$ and $u \in \mathbf{t}U$ equals $(\exists x :: x\mathbf{r}(t,u))$. In the remainder of this section the set $\{x,t,u : t \in \mathbf{t}T \wedge u \in \mathbf{t}U \wedge x\mathbf{r}(t,u) : x\}$ is denoted by S. In view of our mechanistic appreciation it seems reasonable to define the specification of the composite to be $S \lceil (\mathbf{a}T \div \mathbf{a}U)$. We shall prove that this specification is equal to $T\,\mathbf{b}\,U$. To that end we observe the following.

Any trace in $T\,\mathbf{w}\,U$ in which all symbols common to T and U are doubled belongs to S as can be proved by induction. Therefore, $T\,\mathbf{b}\,U \subseteq S \lceil (\mathbf{a}T \div \mathbf{a}U)$. Proving that $S \lceil (\mathbf{a}T \div \mathbf{a}U) \subseteq T\,\mathbf{b}\,U$ is more elaborate. At several places it involves induction. We choose for giving an outline of the proof rather than a fully detailed argument, since the latter would in no way contribute to our understanding of the theory developed in this monograph.

Occurrences of symbols in traces are counted from the left starting from 1. Due to the absence of transmission interference, an odd occurrence of a symbol from $\mathbf{a}T \cap \mathbf{a}U$ in a trace of S originates from the trace structure where this symbol is an output symbol. In the same way we infer that an even occurrence of a common symbol stems from the trace structure where this symbol is an input symbol. Therefore, since the origin of non-common symbols is obvious, an $x \in S$ can uniquely be unravelled into traces $t \in \mathbf{t}T$ and $u \in \mathbf{t}U$ such that $x\mathbf{r}(t,u)$. The unravelling can be effectuated by projecting on a composing trace structure's alphabet and omitting the odd occurrences of a common symbol if it is an input symbol for this trace structure, and the even occurrences in case of an output symbol.

We prove that an arbitrary trace x in S can be transformed, without affecting its projection on $\mathbf{a}T \div \mathbf{a}U$, into a trace in $T\,\mathbf{w}\,U$. As a consequence, $S \lceil (\mathbf{a}T \div \mathbf{a}U)$ is a subset of $T\,\mathbf{b}\,U$, which was the remaining proof obligation. The first step in this transformation is extending x with the common symbols of T and U that occur in x an odd number of times. The resulting trace belongs to S due to the absence of computation interference.

The next step is shifting to the left every even occurrence of a common symbol until it is adjacent to the preceding occurrence of that symbol. In the next paragraphs we show that the resulting trace still belongs to S. Assuming this to hold, we first discuss the final step. Due to steps one and two, all common symbols occur in pairs. Therefore, the unravelling discussed above is the same as projecting on a composing trace structure's alphabet after having replaced each such pair by a single symbol. Hence, this replacement yields a trace in $T\,\mathbf{w}\,U$. In none of the steps have we tampered with the non-common symbols and, hence, $S \lceil (\mathbf{a}S \div \mathbf{a}T)$ is unaffected.

There remains one assumption to be proved, viz. that the trace after shifting still belongs to S. Let $x_0bax_1 \in S$ be such that $a \in \mathbf{a}S \cap \mathbf{a}T$, $b \in \mathbf{a}S \cup \mathbf{a}T$, $a \neq b$, and such that this occurrence of a is even. We prove that x_0abx_1 is an element of S as well. By repeatedly applying this interchange for symbols to be shifted to the left it can be seen that our assumption indeed holds. We distinguish two cases : (i) these occurrences of a and b originate from two distinct trace structures, and (ii) they originate from the same trace structure.

(i) Without loss of generality we assume the unravelling to result in traces $t_0 a t_1 \in \mathbf{t}\,T$ and $u_0 b u_1 \in \mathbf{t}\,U$ such that $x_0 \mathbf{r}(t_0, u_0)$. Since $x_0 ba \in S$, we infer from Definition 4.3 that $b \in \mathbf{o}\,T \Rightarrow \#_b t_0 > \#_b u_0$ and that $a \in \mathbf{o}\,U \Rightarrow \#_a u_0 b > \#_a t_0$. Since $a \neq b$, we derive $a \in \mathbf{o}\,U \Rightarrow \#_a u_0 > \#_a t_0$ and $b \in \mathbf{o}\,T \Rightarrow \#_b t_0 a > \#_b u_0$, which implies $x_0 ab\,\mathbf{r}(t_0 a, u_0 b)$. Moreover, it can be seen from this definition that the construction of x_1 depends on t_1, u_1, and the number of times each symbol occurs in $x_0 ba$ only and, hence, not on the ordering of symbols in $x_0 ba$. This implies that also $x_0 abx_1 \in S$.

(ii) Without loss of generality we assume the unravelling of $x_0 bax_1$ to result in traces $t_0 bat_1 \in \mathbf{t}\,T$ and $u_0 u_1 \in \mathbf{t}\,U$ such that $x_0 \mathbf{r}(t_0, u_0)$. It suffices to prove that $t_0 abt_1 \in \mathbf{t}\,T$, since this implies $x_0 ab \in S$ and since the construction of x_1 does not depend on the ordering of symbols in $x_0 ba$. This occurrence of a is even in $x_0 ba$, hence, $a \in \mathbf{i}\,T \cap \mathbf{o}\,U$. If $b \in \mathbf{i}\,T$ then $t_0 abt_1 \in \mathbf{t}\,T$ on account of $\mathbf{R}_3$. Therefore, assume $b \in \mathbf{o}\,T$. From Definition 4.3 it can be seen that $\#_a u_0 > \#_a t_0 b$, which implies $\#_a u_0 > \#_a t_0$. Due to the absence of computation interference we conclude $t_0 a \in \mathbf{t}\,T$ and, applying $\mathbf{R}_5'''$, $t_0 ab \in \mathbf{t}\,T$. We prove $t_0 abt_2 \in \mathbf{t}\,T$ for an arbitrary prefix t_2 of t_1. For $t_2 = \epsilon$ it is obvious. If $t_2 c$ is a prefix of t_1 such that $t_0 abt_2 \in \mathbf{t}\,T$ then we distinguish the following three cases (using $t_0 bat_2 c \in \mathbf{t}\,T$ on account of the prefix-closedness of $\mathbf{t}\,T$). If $c \in \mathbf{o}\,T$ then $t_0 abt_2 c \in \mathbf{t}\,T$ on account of $\mathbf{R}_4''$. If $c \in \mathbf{i}\,T \setminus (\mathbf{a}\,T \cap \mathbf{a}\,U)$ then $t_0 abt_2 c \in \mathbf{t}\,T$, since $\mathbf{a}\,T \setminus (\mathbf{a}\,T \cap \mathbf{a}\,U)$ is independent with respect to T and $a \notin \mathbf{a}\,T \setminus (\mathbf{a}\,T \cap \mathbf{a}\,U)$. If $c \in \mathbf{i}\,T \cap \mathbf{o}\,U$ then $t_0 abt_2 c \in \mathbf{t}\,T$ on account of the absence of computation interference.

5

Closure properties

In this chapter we discuss the closure of the four classes under composition of connectable trace structures. It turns out that all but $\mathbf{C}_3$ are closed under composition. In a number of examples we apply the theory thus far developed and derive specifications of the composite from the specifications of the composing parts.

We begin this chapter with a section that contains a number of lemmata for trace structures obtained by weaving. Most of these lemmata are counterparts of lemmata in Chapters 2 and 3 on the shifting of symbols. In Section 5.1 we show that a composite obtained by weaving satisfies the rules for delay-insensitivity, provided that the composing parts do. The next section deals with $\mathbf{R}_0$ through $\mathbf{R}_3$ for a composite obtained by blending. In order to prove the $\mathbf{R}_4$'s and $\mathbf{R}_5$'s, which is done in Sections 5.4, 5.5, and 5.6, we need a better understanding of the relation between the weave and the blend of two trace structures. This is explored in Section 5.3. By this exploration the crucial distinction between $\mathbf{C}_2$ and $\mathbf{C}_3$ becomes clear.

In the proofs of this chapter we frequently use the definition of weaving. Part of this definition concerns the domain of the traces considered. For the sake of brevity we omit these domain concerns, appealing to the willingness of the reader to add them at the appropriate places.

5.0. Shifting symbols in trace structures obtained by weaving

The lemmata in this section are counterparts of lemmata in Chapters 2 and 3 on the shifting of symbols. Most of the proofs are merely applications of the corresponding lemmata in these chapters. Therefore, we prove a few lemmata in detail, assuming that this provides a sufficient clue for the derivation of the remaining proofs.

Lemma 5.0 : (cf. Lemma 3.1) For connectable $\mathbf{C}_4$'s S and T, for traces s, t, and u, and for symbol $a \in \mathbf{i}(S \mathbf{w} T)$ such that $t \lceil (\mathbf{a} S \div \mathbf{a} T) = \epsilon$

$$stau \in \mathbf{t}(S \mathbf{w} T) \Rightarrow satu \in \mathbf{t}(S \mathbf{w} T)$$

Proof : By Property 3.0 we assume without loss of generality

$$a \in \mathbf{i}S \setminus \mathbf{a}T \tag{0}$$

Now we derive

$stau \in \mathbf{t}(S \mathbf{w} T)$

$=$ { definition of weaving }

$stau \lceil \mathbf{a}S \in \mathbf{t}S \ \wedge \ stau \lceil \mathbf{a}T \in \mathbf{t}T$

$=$ { distribution of projection over concatenation, using (0) }

$(s \lceil \mathbf{a}S)(t \lceil \mathbf{a}S)a(u \lceil \mathbf{a}S) \in \mathbf{t}S \ \wedge \ stu \lceil \mathbf{a}T \in \mathbf{t}T$

$\Rightarrow$ { since $t \lceil (\mathbf{a}S \div \mathbf{a}T) = \epsilon$, we have $t \lceil \mathbf{a}S \lceil (\mathbf{a}S \setminus \mathbf{a}T) = \epsilon$. Moreover, $\mathbf{a}S \setminus \mathbf{a}T$ is independent with respect to S, due to the connectability of S and T. Hence, we may apply Lemma 3.1 }

$(s \lceil \mathbf{a}S)a(t \lceil \mathbf{a}S)(u \lceil \mathbf{a}S) \in \mathbf{t}S \ \wedge \ stu \lceil \mathbf{a}T \in \mathbf{t}T$

$=$ { distribution of projection over concatenation, using (0) }

$satu \lceil \mathbf{a}S \in \mathbf{t}S \ \wedge \ satu \lceil \mathbf{a}T \in \mathbf{t}T$

$=$ {definition of weaving }

$satu \in \mathbf{t}(S \mathbf{w} T)$

(End of Proof)

In exactly the same way we derive the next lemma.

Lemma 5.1 : (cf. Lemma 3.2) For connectable $\mathbf{C}_4$'s S and T, for traces s, t, and u, and for symbol $a \in \mathbf{o}(S \mathbf{w} T)$ such that $t \lceil (\mathbf{a} S \div \mathbf{a} T) = \epsilon$

$$satu \in \mathbf{t}(S \mathbf{w} T) \Rightarrow stau \in \mathbf{t}(S \mathbf{w} T)$$

From Lemmata 5.0 and 5.1 we derive

Lemma 5.2 : (cf. Lemma 3.3) For connectable C_4's S and T, for traces s, t, and u, and for symbols $a \in \mathbf{a}S \div \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$ of the same type such that $t\lceil(\mathbf{a}S \div \mathbf{a}T) = \epsilon$

$$satbu \in \mathbf{t}(S \mathbf{w} T) \Rightarrow sabtu \in \mathbf{t}(S \mathbf{w} T) \lor stabu \in \mathbf{t}(S \mathbf{w} T)$$

Lemma 5.3 : (cf. Lemma 2.8) For connectable C_2's S and T, for traces s and t, and for symbol $a \in \mathbf{a}S \cap \mathbf{a}T$ such that $t\lceil\{a\} = \epsilon$

$$sa \in \mathbf{t}(S \mathbf{w} T) \land st \in \mathbf{t}(S \mathbf{w} T) \Rightarrow sta \in \mathbf{t}(S \mathbf{w} T)$$

Proof : Without loss of generality we assume

$$a \in \mathbf{o}S \cap \mathbf{i}T \qquad (0)$$

We prove that the left-hand side implies (i) $sta\lceil\mathbf{a}S \in \mathbf{t}S$ and (ii) $sta\lceil\mathbf{a}T \in \mathbf{t}T$, which implies, by the definition of weaving, the right-hand side.

(i) $sta\lceil\mathbf{a}S \in \mathbf{t}S$

$sa \in \mathbf{t}(S \mathbf{w} T) \land st \in \mathbf{t}(S \mathbf{w} T)$

$\Rightarrow$ { calculus and definition of weaving }

$sa\lceil\mathbf{a}S \in \mathbf{t}S \land st\lceil\mathbf{a}S \in \mathbf{t}S$

$=$ { distribution of projection over concatenation, using (0) }

$(s\lceil\mathbf{a}S)a \in \mathbf{t}S \land (s\lceil\mathbf{a}S)(t\lceil\mathbf{a}S) \in \mathbf{t}S$

$\Rightarrow$ { from $t\lceil\{a\} = \epsilon$ we infer $t\lceil\mathbf{a}S\lceil\{a\} = \epsilon$. Hence, since $a \in \mathbf{o}S$, we may apply Lemma 2.8 }

$(s\lceil\mathbf{a}S)(t\lceil\mathbf{a}S)a \in \mathbf{t}S$

$=$ { distribution of projection over concatenation, using (0) }

$sta\lceil\mathbf{a}S \in \mathbf{t}S$

(ii) $sta\lceil\mathbf{a}T \in \mathbf{t}T$

$sa \in \mathbf{t}(S \mathbf{w} T) \land st \in \mathbf{t}(S \mathbf{w} T)$

$\Rightarrow$ { calculus and definition of weaving, and (i) }

$st\lceil\mathbf{a}T \in \mathbf{t}T \land sta\lceil\mathbf{a}S \in \mathbf{t}S$

$\Rightarrow$ { distribution of projection over concatenation, using (0), and projection on $\mathbf{a}S \cap \mathbf{a}T$ }

$st\lceil\mathbf{a}T \in \mathbf{t}T \land (st\lceil\mathbf{a}S)a\lceil(\mathbf{a}S \cap \mathbf{a}T) \in \mathbf{t}S\lceil(\mathbf{a}S \cap \mathbf{a}T)$

$=$ { Property 1.1, using $\mathbf{a}S \cap \mathbf{a}T \subseteq \mathbf{a}S$ and $\mathbf{a}S \cap \mathbf{a}T \subseteq \mathbf{a}T$ }
$st \lceil \mathbf{a}T \in \mathbf{t}T \wedge (st \lceil \mathbf{a}T)a \lceil (\mathbf{a}S \cap \mathbf{a}T) \in \mathbf{t}S \lceil (\mathbf{a}S \cap \mathbf{a}T)$
$=$ { $\mathbf{t}S \lceil (\mathbf{a}S \cap \mathbf{a}T) = \mathbf{t}T \lceil (\mathbf{a}S \cap \mathbf{a}T)$, since S and T are connectable }
$st \lceil \mathbf{a}T \in \mathbf{t}T \wedge (st \lceil \mathbf{a}T)a \lceil (\mathbf{a}S \cap \mathbf{a}T) \in \mathbf{t}T \lceil (\mathbf{a}S \cap \mathbf{a}T)$
$=$ { $\mathbf{a}S \cap \mathbf{a}T$ is independent with respect to T and $a \in (\mathbf{a}S \cap \mathbf{a}T) \cap \mathbf{i}T$ according to (0) }
$(st \lceil \mathbf{a}T)a \in \mathbf{t}T$
$=$ { distribution of projection over concatenation, using (0) }
$sta \lceil \mathbf{a}T \in \mathbf{t}T$

(End of Proof)

Notice that we used here explicitly, as we will in the next proof as well, the last requirement for connectability, viz. $S \lceil (\mathbf{a}S \cap \mathbf{a}T) = T \lceil (\mathbf{a}S \cap \mathbf{a}T)$.

Lemma 5.4 : (cf. Lemma 2.9) For connectable $\mathbf{C}_2$'s S and T, for traces s, t, and u, and for symbol $a \in \mathbf{a}S \cap \mathbf{a}T$ such that $t \lceil \{ a \} = \epsilon$

$$sa \in \mathbf{t}(S \mathbf{w} T) \wedge stau \in \mathbf{t}(S \mathbf{w} T) \Rightarrow satu \in \mathbf{t}(S \mathbf{w} T)$$

Proof : Without loss of generality we assume

$$a \in \mathbf{o}S \cap \mathbf{i}T \qquad (0)$$

Again we prove $satu \lceil \mathbf{a}S \in \mathbf{t}S$ and $satu \lceil \mathbf{a}T \in \mathbf{t}T$ separately.

(i) $satu \lceil \mathbf{a}S \in \mathbf{t}S$

$sa \in \mathbf{t}(S \mathbf{w} T) \wedge stau \in \mathbf{t}(S \mathbf{w} T)$
$\Rightarrow$ { definition of weaving and calculus }
$sa \lceil \mathbf{a}S \in \mathbf{t}S \wedge stau \lceil \mathbf{a}S \in \mathbf{t}S$
$=$ { distribution of projection over concatenation, using (0) }
$(s \lceil \mathbf{a}S)a \in \mathbf{t}S \wedge (s \lceil \mathbf{a}S)(t \lceil \mathbf{a}S)a(u \lceil \mathbf{a}S) \in \mathbf{t}S$
$\Rightarrow$ { from $t \lceil \{ a \} = \epsilon$ we infer $t \lceil \mathbf{a}S \lceil \{ a \} = \epsilon$. Hence, since $a \in \mathbf{o}S$, we may apply Lemma 2.10 }
$(\forall w_0, w_1 : w_0 w_1 = t \lceil \mathbf{a}S : (s \lceil \mathbf{a}S) w_0 a w_1 (u \lceil \mathbf{a}S) \in \mathbf{t}S)$ (1)
$\Rightarrow$ {instantiation }
$(s \lceil \mathbf{a}S)a(t \lceil \mathbf{a}S)(u \lceil \mathbf{a}S) \in \mathbf{t}S$

$=$ { distribution of projection over concatenation, using (0) }
$satu \lceil \mathbf{a}S \in \mathbf{t}S$

(ii) $satu \lceil \mathbf{a}T \in \mathbf{t}T$

$sa \in \mathbf{t}(S \,\mathbf{w}\, T) \wedge stau \in \mathbf{t}(S \,\mathbf{w}\, T)$
$\Rightarrow$ { definition of weaving, (1), and calculus }
$stau \lceil \mathbf{a}T \in \mathbf{t}T \wedge (\forall w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}S : (s \lceil \mathbf{a}S)w_0aw_1(u \lceil \mathbf{a}S) \in \mathbf{t}S)$
$\Rightarrow$ { $\mathbf{t}S$ is prefix-closed and projection on $\mathbf{a}S \cap \mathbf{a}T$ }
$stau \lceil \mathbf{a}T \in \mathbf{t}T \wedge$
$(\forall w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}S : (s \lceil \mathbf{a}S)w_0a \lceil (\mathbf{a}S \cap \mathbf{a}T) \in \mathbf{t}S \lceil (\mathbf{a}S \cap \mathbf{a}T))$
$=$ { $\mathbf{t}S \lceil (\mathbf{a}S \cap \mathbf{a}T) = \mathbf{t}T \lceil (\mathbf{a}S \cap \mathbf{a}T)$, since S and T are connectable; distribution of projection over concatenation, using (0) and Property 1.1 }
$(s \lceil \mathbf{a}T)(t \lceil \mathbf{a}T)a(u \lceil \mathbf{a}T) \in \mathbf{t}T \wedge$
$(\forall w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}S : (s \lceil (\mathbf{a}S \cap \mathbf{a}T))(w_0 \lceil (\mathbf{a}S \cap \mathbf{a}T))a$
$\in \mathbf{t}T \lceil (\mathbf{a}S \cap \mathbf{a}T))$
$=$ { the set $\{ w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}S : w_0 \lceil (\mathbf{a}S \cap \mathbf{a}T) \}$ equals the set $\{ w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}T : w_0 \lceil (\mathbf{a}S \cap \mathbf{a}T) \}$ }
$(s \lceil \mathbf{a}T)(t \lceil \mathbf{a}T)a(u \lceil \mathbf{a}T) \in \mathbf{t}T \wedge$
$(\forall w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}T : (s \lceil (\mathbf{a}S \cap \mathbf{a}T))(w_0 \lceil (\mathbf{a}S \cap \mathbf{a}T))a$
$\in \mathbf{t}T \lceil (\mathbf{a}S \cap \mathbf{a}T))$
$=$ { distribution of projection over concatenation, using (0) and Property 1.1; $\mathbf{t}T$ is prefix-closed }
$(s \lceil \mathbf{a}T)(t \lceil \mathbf{a}T)a(u \lceil \mathbf{a}T) \in \mathbf{t}T \wedge$
$(\forall w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}T : (s \lceil \mathbf{a}T)w_0 \in \mathbf{t}T \wedge$
$(s \lceil \mathbf{a}T)w_0a \lceil (\mathbf{a}S \cap \mathbf{a}T) \in \mathbf{t}T \lceil (\mathbf{a}S \cap \mathbf{a}T))$
$=$ { $\mathbf{a}S \cap \mathbf{a}T$ is independent with respect to T, since S and T are connectable. Moreover, $a \in (\mathbf{a}S \cap \mathbf{a}T) \cap \mathbf{i}T$ according to (0) }
$(s \lceil \mathbf{a}T)(t \lceil \mathbf{a}T)a(u \lceil \mathbf{a}T) \in \mathbf{t}T \wedge$
$(\forall w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}T : (s \lceil \mathbf{a}T)w_0a \in \mathbf{t}T)$
$\Rightarrow$ { Lemma 2.11, since $a \in \mathbf{i}T$ according to (0) }
$(\forall w_0, w_1 : w_0w_1 = t \lceil \mathbf{a}T : (s \lceil \mathbf{a}T)w_0aw_1(u \lceil \mathbf{a}T) \in \mathbf{t}T)$
$\Rightarrow$ { instantiation and distribution of projection over concatenation, using (0) }
$satu \lceil \mathbf{a}T \in \mathbf{t}T$

(End of Proof)

On account of Lemmata 5.3 and 5.4 we may, given two connectable $\mathbf{C}_2$'s S and T, symbol $a \in \mathbf{a}S \cap \mathbf{a}T$, and traces sa and st in $\mathbf{t}(S \mathbf{w} T)$, shift the left-most a in t to the left of t, or, if no such a exists, insert an a between s and t. Therefore, the following corollary is a straightforward application of these two lemmata.

Corollary 5.0 : For connectable $\mathbf{C}_2$'s S and T, for traces s, t, and u, and for symbols $a \in \mathbf{a}S \cap \mathbf{a}T$ and b such that $b \neq a$

$$\begin{aligned} & sa \in \mathbf{t}(S \mathbf{w} T) \wedge stbu \in \mathbf{t}(S \mathbf{w} T) \Rightarrow \\ & (\exists w_0, w_1 :: saw_0bw_1 \in \mathbf{t}(S \mathbf{w} T) \wedge w_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = t \lceil (\mathbf{a}S \div \mathbf{a}T) \\ & \qquad \wedge w_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = u \lceil (\mathbf{a}S \div \mathbf{a}T) \wedge \mathbf{l}w_0 \leqslant \mathbf{l}t) \end{aligned}$$

We conclude this section with two lemmata which are quite similar to Lemmata 5.3 and 5.4 but much easier to prove. The distinction is that symbol a is an element of $\mathbf{o}(S \mathbf{w} T)$ rather than of $\mathbf{a}S \cap \mathbf{a}T$.

Lemma 5.5 : (cf. Lemma 2.8) For connectable $\mathbf{C}_2$'s S and T, for traces s and t, and for symbol $a \in \mathbf{o}(S \mathbf{w} T)$ such that $t \lceil \{ a \} = \epsilon$

$$sa \in \mathbf{t}(S \mathbf{w} T) \wedge st \in \mathbf{t}(S \mathbf{w} T) \Rightarrow sta \in \mathbf{t}(S \mathbf{w} T)$$

Lemma 5.6 : (cf. Lemma 2.9) For connectable $\mathbf{C}_2$'s S and T, for traces s, t, and u, and for symbol $a \in \mathbf{o}(S \mathbf{w} T)$ such that $t \lceil \{ a \} = \epsilon$

$$sa \in \mathbf{t}(S \mathbf{w} T) \wedge stau \in \mathbf{t}(S \mathbf{w} T) \Rightarrow satu \in \mathbf{t}(S \mathbf{w} T)$$

5.1. $\mathbf{R}_2$ through $\mathbf{R}_5$ for trace structures obtained by weaving

In this section we show that $\mathbf{R}_2$ through $\mathbf{R}_5$ hold for the composite obtained by weaving of connectable trace structures. Most of the proofs merely require a frequent use of distribution of projection over concatenation and of the definitions of weaving and the four classes. Therefore, we prove only some of the lemmata.

Lemma 5.7 : (cf. $\mathbf{R}_2$) For connectable $\mathbf{C}_4$'s S and T, for trace s, and for symbol a

$$saa \notin \mathbf{t}(S \mathbf{w} T)$$

Lemma 5.8 : (cf. $\mathbf{R}_3$) For connectable $\mathbf{C}_4$'s S and T, for traces s and t, and for symbols $a \in \mathbf{a}S \div \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$ of the same type

$$sabt \in \mathbf{t}(S \mathbf{w} T) \; = \; sbat \in \mathbf{t}(S \mathbf{w} T)$$

Proof : Without loss of generality we assume $a \in \mathbf{a}S$. We distinguish two cases (i) $b \in \mathbf{a}S$, and (ii) $b \notin \mathbf{a}S$.

(i) Since $a \in \mathbf{a}S \div \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$ we have in this case

$$a \in \mathbf{a}S \setminus \mathbf{a}T \quad \text{and} \quad b \in \mathbf{a}S \setminus \mathbf{a}T \tag{0}$$

Now we derive

$$\begin{array}{ll}
 & sabt \in \mathbf{t}(S \mathbf{w} T) \\
= & \{ \text{ definition of weaving } \} \\
 & sabt \lceil \mathbf{a}S \in \mathbf{t}S \;\wedge\; sabt \lceil \mathbf{a}T \in \mathbf{t}T \\
= & \{ \text{ distribution of projection over concatenation, using (0) } \} \\
 & (s \lceil \mathbf{a}S)ab(t \lceil \mathbf{a}S) \in \mathbf{t}S \;\wedge\; st \lceil \mathbf{a}T \in \mathbf{t}T \\
= & \{ \; \mathbf{R}_3, \text{ since } S \text{ is a } \mathbf{C}_4 \text{ and } a \text{ and } b \text{ are of the same type } \} \\
 & (s \lceil \mathbf{a}S)ba(t \lceil \mathbf{a}S) \in \mathbf{t}S \;\wedge\; st \lceil \mathbf{a}T \in \mathbf{t}T \\
= & \{ \text{ distribution of projection over concatenation, using (0) } \} \\
 & sbat \lceil \mathbf{a}S \in \mathbf{t}S \;\wedge\; sbat \lceil \mathbf{a}T \in \mathbf{t}T \\
= & \{ \text{ definition of weaving } \} \\
 & sbat \in \mathbf{t}(S \mathbf{w} T)
\end{array}$$

(ii) In this case we have $a \in \mathbf{a}S \setminus \mathbf{a}T$ and $b \in \mathbf{a}T \setminus \mathbf{a}S$. Now Property 1.3 yields the result desired.

(End of Proof)

Lemma 5.9 : (cf. $\mathbf{R}_4'$) For connectable $\mathbf{C}_3$'s S and T, for traces s and t, and for symbols $a \in \mathbf{a}S \div \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$ of different types

$$sa \in \mathbf{t}(S \mathbf{w} T) \wedge sbat \in \mathbf{t}(S \mathbf{w} T) \Rightarrow sabt \in \mathbf{t}(S \mathbf{w} T)$$

Lemma 5.10 : (cf. $\mathbf{R}_4''$) For connectable $\mathbf{C}_4$'s S and T, for traces s and t, and for symbols $a \in \mathbf{a}S \div \mathbf{a}T$, $b \in \mathbf{a}S \div \mathbf{a}T$, and $c \in \mathbf{a}S \div \mathbf{a}T$ such that b is of another type than a and c

$$sabtc \in \mathbf{t}(S \mathbf{w} T) \wedge sbat \in \mathbf{t}(S \mathbf{w} T) \Rightarrow sbatc \in \mathbf{t}(S \mathbf{w} T)$$

Proof : We distinguish three cases : (i) a, b, and c belong to the same trace structure, (ii) c belongs to another trace structure than a and b, and (iii) a and b belong to different trace structures.

(i) a, b, and c belong to the same trace structure. Without loss of generality we assume this trace structure to be S. Hence,

$$a \in \mathbf{a}S \setminus \mathbf{a}T,\ \ b \in \mathbf{a}S \setminus \mathbf{a}T,\ \text{ and }\ c \in \mathbf{a}S \setminus \mathbf{a}T \tag{0}$$

Now we derive

$sabtc \in \mathbf{t}(S \mathbf{w} T) \wedge sbat \in \mathbf{t}(S \mathbf{w} T)$
$=$ { definition of weaving }
$sabtc \lceil \mathbf{a}S \in \mathbf{t}S \wedge sabtc \lceil \mathbf{a}T \in \mathbf{t}T \wedge sbat \lceil \mathbf{a}S \in \mathbf{t}S \wedge sbat \lceil \mathbf{a}T \in \mathbf{t}T$
$=$ { distribution of projection over concatenation, using (0); calculus }
$(s \lceil \mathbf{a}S)ab(t \lceil \mathbf{a}S)c \in \mathbf{t}S \wedge (s \lceil \mathbf{a}S)ba(t \lceil \mathbf{a}S) \in \mathbf{t}S \wedge st \lceil \mathbf{a}T \in \mathbf{t}T$
$\Rightarrow$ { $\mathbf{R}_4''$ }
$(s \lceil \mathbf{a}S)ba(t \lceil \mathbf{a}S)c \in \mathbf{t}S \wedge st \lceil \mathbf{a}T \in \mathbf{t}T$
$=$ { distribution of projection over concatenation, using (0) }
$sbatc \lceil \mathbf{a}S \in \mathbf{t}S \wedge sbatc \lceil \mathbf{a}T \in \mathbf{t}T$
$=$ { definition of weaving }
$sbatc \in \mathbf{t}(S \mathbf{w} T)$

(ii) c belongs to another trace structure than a and b. Without loss of generality we assume $c \in \mathbf{a}T$. Hence,

$$a \in \mathbf{a}S \setminus \mathbf{a}T,\ \ b \in \mathbf{a}S \setminus \mathbf{a}T,\ \text{ and }\ c \in \mathbf{a}T \setminus \mathbf{a}S \tag{1}$$

Now we derive

$sabtc \in \mathbf{t}(S \mathbf{w} T) \wedge sbat \in \mathbf{t}(S \mathbf{w} T)$
$=$ { definition of weaving }
$sabtc \lceil \mathbf{a} S \in \mathbf{t} S \wedge sabtc \lceil \mathbf{a} T \in \mathbf{t} T \wedge sbat \lceil \mathbf{a} S \in \mathbf{t} S \wedge sbat \lceil \mathbf{a} T \in \mathbf{t} T$
$\Rightarrow$ { calculus and distribution of projection over concatenation, using (1) }
$sbatc \lceil \mathbf{a} S \in \mathbf{t} S \wedge sbatc \lceil \mathbf{a} T \in \mathbf{t} T$
$=$ { definition of weaving }
$sbatc \in \mathbf{t}(S \mathbf{w} T)$

(iii) a and b belong to different trace structures. Then, according to Property 1.3, $sabtc \in \mathbf{t}(S \mathbf{w} T) = sbatc \in \mathbf{t}(S \mathbf{w} T)$.
(End of Proof)

Lemma 5.11 : (cf. $\mathbf{R}_4''$) For connectable $\mathbf{C}_4$'s S and T, for traces s and t, and for symbols $a \in \mathbf{o}(S \mathbf{w} T)$, $b \in \mathbf{i}(S \mathbf{w} T)$, and $c \in \mathbf{a} S \cap \mathbf{a} T$

$$sabtc \in \mathbf{t}(S \mathbf{w} T) \wedge sbat \in \mathbf{t}(S \mathbf{w} T) \Rightarrow sbatc \in \mathbf{t}(S \mathbf{w} T)$$

Proof : We distinguish two cases : (i) a and b belong to the same trace structure, and (ii) a and b belong to different trace structures.

(i) a and b belong to the same trace structure. Without loss of generality we assume this trace structure to be S. Hence,

$$a \in \mathbf{o} S \setminus \mathbf{a} T \quad \text{and} \quad b \in \mathbf{i} S \setminus \mathbf{a} T \tag{0}$$

Next, we distinguish (a) $c \in \mathbf{i} S \cap \mathbf{o} T$, and (b) $c \in \mathbf{o} S \cap \mathbf{i} T$

(a) $c \in \mathbf{i} S \cap \mathbf{o} T$. Now we derive

$sabtc \in \mathbf{t}(S \mathbf{w} T) \wedge sbat \in \mathbf{t}(S \mathbf{w} T)$
$=$ { definition of weaving }
$sabtc \lceil \mathbf{a} S \in \mathbf{t} S \wedge sabtc \lceil \mathbf{a} T \in \mathbf{t} T \wedge sbat \lceil \mathbf{a} S \in \mathbf{t} S \wedge sbat \lceil \mathbf{a} T \in \mathbf{t} T$
$\Rightarrow$ { distribution of projection over concatenation, using (0). Moreover, projection on $\mathbf{a} S \cap \mathbf{a} T$, using Property 1.1, $c \in \mathbf{a} S \cap \mathbf{a} T$, and (0) }
$(s \lceil \mathbf{a} S) ba (t \lceil \mathbf{a} S) c \lceil (\mathbf{a} S \cap \mathbf{a} T) \in \mathbf{t} S \lceil (\mathbf{a} S \cap \mathbf{a} T) \wedge stc \lceil \mathbf{a} T \in \mathbf{t} T$
$\wedge (s \lceil \mathbf{a} S) ba (t \lceil \mathbf{a} S) \in \mathbf{t} S$

$= \{ \mathbf{a}S \cap \mathbf{a}T$ is independent with respect to S and $c \in (\mathbf{a}S \cap \mathbf{a}T) \cap \mathbf{i}S \}$

$(s \lceil \mathbf{a}S)ba(t \lceil \mathbf{a}S)c \in \mathbf{t}S \ \wedge\ stc \lceil \mathbf{a}T \in \mathbf{t}T$

$= \{$ distribution of projection over concatenation, using (0) and $c \in \mathbf{a}S \}$

$sbatc \lceil \mathbf{a}S \in \mathbf{t}S \ \wedge\ sbatc \lceil \mathbf{a}T \in \mathbf{t}T$

$= \{$ definition of weaving $\}$

$sbatc \in \mathbf{t}(S\ \mathbf{w}\ T)$

(b) $c \in \mathbf{o}S \cap \mathbf{i}T$

$sabtc \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ sbat \in \mathbf{t}(S\ \mathbf{w}\ T)$

$= \{$ definition of weaving $\}$

$sabtc \lceil \mathbf{a}S \in \mathbf{t}S \ \wedge\ sabtc \lceil \mathbf{a}T \in \mathbf{t}T \ \wedge\ sbat \lceil \mathbf{a}S \in \mathbf{t}S \ \wedge\ sbat \lceil \mathbf{a}T \in \mathbf{t}T$

$\Rightarrow \{$ distribution of projection over concatenation, using (0) and $c \in \mathbf{a}S \}$

$(s \lceil \mathbf{a}S)ab(t \lceil \mathbf{a}S)c \in \mathbf{t}S \ \wedge\ stc \lceil \mathbf{a}T \in \mathbf{t}T \ \wedge\ (s \lceil \mathbf{a}S)ba(t \lceil \mathbf{a}S) \in \mathbf{t}S$

$\Rightarrow \{\mathbf{R}_4''$, since $a \in \mathbf{o}S$ and $b \in \mathbf{i}S$ according to (0), and $c \in \mathbf{o}S \}$

$(s \lceil \mathbf{a}S)ba(t \lceil \mathbf{a}S)c \in \mathbf{t}S \ \wedge\ stc \lceil \mathbf{a}T \in \mathbf{t}T$

$= \{$ distribution of projection over concatenation, using (0) and $c \in \mathbf{a}S \}$

$sbatc \lceil \mathbf{a}S \in \mathbf{t}S \ \wedge\ sbatc \lceil \mathbf{a}T \in \mathbf{t}T$

$= \{$ definition of weaving $\}$

$sbatc \in \mathbf{t}(S\ \mathbf{w}\ T)$

(ii) a and b belong to different trace structures. Then, according to Property 1.3, $sabtc \in \mathbf{t}(S\ \mathbf{w}\ T) = sbatc \in \mathbf{t}(S\ \mathbf{w}\ T)$.

(End of Proof)

Lemma 5.12 : (cf. $\mathbf{R}_5'$) For connectable $\mathbf{C}_1$'s S and T, for trace s, and for distinct symbols $a \in \mathbf{a}S \div \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$

$$sa \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ sb \in \mathbf{t}(S\ \mathbf{w}\ T) \Rightarrow sab \in \mathbf{t}(S\ \mathbf{w}\ T)$$

Lemma 5.13 : (cf. $\mathbf{R}_5''$) For connectable $\mathbf{C}_2$'s S and T, for trace s, and for distinct symbols $a \in \mathbf{a}S \div \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$, not both belonging to $\mathbf{i}(S\ \mathbf{w}\ T)$

$$sa \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ sb \in \mathbf{t}(S\ \mathbf{w}\ T) \Rightarrow sab \in \mathbf{t}(S\ \mathbf{w}\ T)$$

Lemma 5.14 : (cf. $\mathbf{R}_5'''$) For connectable $\mathbf{C}_4$'s S and T, for trace s, and for symbols $a \in \mathbf{a}S \div \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$ of different types

$$sa \in \mathbf{t}(S \mathbf{w} T) \wedge sb \in \mathbf{t}(S \mathbf{w} T) \Rightarrow sab \in \mathbf{t}(S \mathbf{w} T)$$

5.2. $\mathbf{R}_0$ through $\mathbf{R}_3$ for trace structures obtained by blending

Using the lemmata derived in the preceding section it is easy to prove $\mathbf{R}_0$ through $\mathbf{R}_3$ for the composite of two connectable $\mathbf{C}_4$'s. The proofs are short and straightforward and, therefore, all are omitted but one.

Lemma 5.15 : For connectable $\mathbf{C}_4$'s S and T

0) $\mathbf{i}(S \mathbf{b} T) \cup \mathbf{o}(S \mathbf{b} T) = \mathbf{a}(S \mathbf{b} T)$
1) $\mathbf{t}(S \mathbf{b} T)$ is prefix-closed and non-empty
2) for trace s and symbol $a \in \mathbf{a}(S \mathbf{b} T)$ $\quad saa \notin \mathbf{t}(S \mathbf{b} T)$
3) for traces s and t, and for symbols $a \in \mathbf{a}(S \mathbf{b} T)$ and $b \in \mathbf{a}(S \mathbf{b} T)$ of the same type $\quad sabt \in \mathbf{t}(S \mathbf{b} T) = sbat \in \mathbf{t}(S \mathbf{b} T)$

Proof of 3) :

$sabt \in \mathbf{t}(S \mathbf{b} T)$

$=$ { definition of blending, using $a \in \mathbf{a}(S \mathbf{b} T)$ and $b \in \mathbf{a}(S \mathbf{b} T)$ }

$(\exists t_0, t_1, t_2 :: t_0 a t_1 b t_2 \in \mathbf{t}(S \mathbf{w} T) \wedge t_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s \wedge t_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = \epsilon$
$\quad \wedge t_2 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { Lemma 5.2, since a and b are of the same type }

$(\exists t_0, t_1, t_2 :: (t_0 ab t_1 t_2 \in \mathbf{t}(S \mathbf{w} T) \vee t_0 t_1 ab t_2 \in \mathbf{t}(S \mathbf{w} T)) \wedge$
$\quad t_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s \wedge t_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = \epsilon \wedge t_2 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$

$=$ { Lemma 5.8 }

$(\exists t_0, t_1, t_2 :: (t_0 ba t_1 t_2 \in \mathbf{t}(S \mathbf{w} T) \vee t_0 t_1 ba t_2 \in \mathbf{t}(S \mathbf{w} T)) \wedge$
$\quad t_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s \wedge t_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = \epsilon \wedge t_2 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { definition of blending, using $a \in \mathbf{a}(S \mathbf{b} T)$ and $b \in \mathbf{a}(S \mathbf{b} T)$ }

$sbat \in \mathbf{t}(S \mathbf{b} T)$

Hence, $sabt \in \mathbf{t}(S \mathbf{b} T) \Rightarrow sbat \in \mathbf{t}(S \mathbf{b} T)$ for symbols a and b of the same type. Therefore, the implication may be replaced by equality.

(End of Proof)

5.3. Internal communications for a blend

The remaining rules to be proved for the blend of two connectable trace structures are less easily derived from those for the weave. The reason is that in the left-hand sides of the implications in these rules the same trace occurs twice. By the standard conversion from an expression in terms of the blend to an expression in terms of the weave, these occurrences convert to possibly distinct traces. As a consequence, the lemmata derived in Section 5.1 are not readily applicable. Therefore, we prove in this section three lemmata that relate traces in the blend to traces in the weave in such a way that we can apply the lemmata derived in Section 5.1. Due to the absence of arbitration in the internal communications, we can prove for $\mathbf{C}_2$'s a stronger lemma than for $\mathbf{C}_4$'s.

Lemma 5.16 : For connectable $\mathbf{C}_2$'s S and T, for traces s, t, and u, and for symbols $a \in \mathbf{a}S \div \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$

$$\begin{aligned} &sat \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sbu \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow \\ &(\exists s_0, s_1, s_2 :: s_0as_1 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0bs_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \\ &\qquad\qquad \wedge s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = t \wedge s_2\lceil(\mathbf{a}S \div \mathbf{a}T) = u) \end{aligned}$$

Proof : We prove by mathematical induction on $\mathbf{l}r_1 + \mathbf{l}r_3$ that for traces r_0, r_1, r_2, r_3, and r_4, such that

$$r_1\lceil(\mathbf{a}S \div \mathbf{a}T) = r_3\lceil(\mathbf{a}S \div \mathbf{a}T) \qquad (0)$$

we have

$$\begin{aligned} &r_0r_1ar_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge r_0r_3br_4 \in \mathbf{t}(S\,\mathbf{w}\,T) \Rightarrow \\ &(\exists s_0, s_1, s_2 :: r_0s_0as_1 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge r_0s_0bs_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge \\ &\quad s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = r_1\lceil(\mathbf{a}S \div \mathbf{a}T) \wedge s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = r_2\lceil(\mathbf{a}S \div \mathbf{a}T) \\ &\quad \wedge s_2\lceil(\mathbf{a}S \div \mathbf{a}T) = r_4\lceil(\mathbf{a}S \div \mathbf{a}T)) \end{aligned} \qquad (1)$$

By choosing $r_0 = \epsilon$ we then have proved the theorem, since

$$\begin{aligned} &sat \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sbu \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow \\ &(\exists r_1, r_2, r_3, r_4 :: r_1ar_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge r_3br_4 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge r_1\lceil(\mathbf{a}S \div \mathbf{a}T) = s \\ &\qquad \wedge r_2\lceil(\mathbf{a}S \div \mathbf{a}T) = t \wedge r_3\lceil(\mathbf{a}S \div \mathbf{a}T) = s \wedge r_4\lceil(\mathbf{a}S \div \mathbf{a}T) = u) \end{aligned}$$

Base : $\mathbf{l}r_1 + \mathbf{l}r_3 = 0$. (1) holds obviously in this case.

Step : Given integer k, $k > 0$. We assume (1) to hold for traces r_0, r_1, r_2, r_3, and r_4 such that $\mathbf{l}r_1 + \mathbf{l}r_3 < k$ and (0). For traces r_0, r_1, r_2, r_3, and r_4 such that (0) and such that

$$\mathbf{l}r_1 + \mathbf{l}r_3 = k \tag{2}$$

we prove (1) in the following way.

We distinguish two cases : (i) r_1 and r_3 start with the same symbol, and (ii) r_1 and r_3 do not start with the same symbol.

(i) r_1 and r_3 start with the same symbol, say $r_1 = cr_5$ and $r_3 = cr_6$. Then we may apply (1) with its r_0, r_1, and r_3 replaced by r_0c, r_5, and r_6 respectively, since $\mathbf{l}r_5 + \mathbf{l}r_6 < \mathbf{l}r_1 + \mathbf{l}r_3$ ($= k$), and since we infer from (0) and the distribution of projection over concatenation $r_5\lceil(\mathbf{a}S \div \mathbf{a}T) = r_6\lceil(\mathbf{a}S \div \mathbf{a}T)$. Now (1) follows by a simple renaming.

(ii) r_1 and r_3 do not start with the same symbol. Moreover, they are not both equal to ϵ according to (2) and the fact that $k > 0$. Hence, at least one of them starts with a symbol of $\mathbf{a}S \cap \mathbf{a}T$, since $r_1\lceil(\mathbf{a}S \div \mathbf{a}T) = r_3\lceil(\mathbf{a}S \div \mathbf{a}T)$ according to (0). Without loss of generality we assume r_1 to start with a symbol of $\mathbf{a}S \cap \mathbf{a}T$, say

$$r_1 = cr_5 \ \text{ and } \ c \in \mathbf{a}S \cap \mathbf{a}T \tag{3}$$

Now we derive

$r_0r_1ar_2 \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge r_0r_3br_4 \in \mathbf{t}(S \mathbin{\mathbf{w}} T)$

$=$ { (3) and the prefix-closedness of $\mathbf{t}(S \mathbin{\mathbf{w}} T)$ }

$r_0cr_5ar_2 \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge r_0c \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge r_0r_3br_4 \in \mathbf{t}(S \mathbin{\mathbf{w}} T)$

$\Rightarrow$ { Corollary 5.0, using $c \in \mathbf{a}S \cap \mathbf{a}T$ and $b \in \mathbf{a}S \div \mathbf{a}T$ }

$(\exists w_0, w_1 :: r_0cr_5ar_2 \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge r_0cw_0bw_1 \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge \mathbf{l}w_0 \leqslant \mathbf{l}r_3 \wedge$
$\quad w_0\lceil(\mathbf{a}S \div \mathbf{a}T) = r_3\lceil(\mathbf{a}S \div \mathbf{a}T) \wedge w_1\lceil(\mathbf{a}S \div \mathbf{a}T) = r_4\lceil(\mathbf{a}S \div \mathbf{a}T))$

$\Rightarrow$ { $r_3\lceil(\mathbf{a}S \div \mathbf{a}T) = r_5\lceil(\mathbf{a}S \div \mathbf{a}T)$ on account of (0) and (3). Moreover, for trace w_0 with $\mathbf{l}w_0 \leqslant \mathbf{l}r_3$ we derive $\mathbf{l}w_0 + \mathbf{l}r_5 < k$ on account of (2) and (3) }

$(\exists w_0, w_1 :: r_0cr_5ar_2 \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge r_0cw_0bw_1 \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge \mathbf{l}w_0 + \mathbf{l}r_5 < k \wedge$
$\quad w_0\lceil(\mathbf{a}S \div \mathbf{a}T) = r_5\lceil(\mathbf{a}S \div \mathbf{a}T) \wedge w_1\lceil(\mathbf{a}S \div \mathbf{a}T) = r_4\lceil(\mathbf{a}S \div \mathbf{a}T))$

$\Rightarrow$ { (1), applicable on account of the induction hypothesis }

$(\exists w_0, w_1, s_0, s_1, s_2 :: r_0cs_0as_1 \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge r_0cs_0bs_2 \in \mathbf{t}(S \mathbin{\mathbf{w}} T) \wedge$
$\quad s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = r_5\lceil(\mathbf{a}S \div \mathbf{a}T) \wedge s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = r_2\lceil(\mathbf{a}S \div \mathbf{a}T) \wedge$

$$s_2\lceil(\mathbf{a}S \div \mathbf{a}T) = w_1\lceil(\mathbf{a}S \div \mathbf{a}T) \wedge w_1\lceil(\mathbf{a}S \div \mathbf{a}T) = r_4\lceil(\mathbf{a}S \div \mathbf{a}T))$$

$\Rightarrow$ { calculus and renaming cs_0, using (3) }

$$(\exists s_0, s_1, s_2 :: r_0 s_0 a s_1 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge r_0 s_0 b s_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge$$
$$s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = r_1\lceil(\mathbf{a}S \div \mathbf{a}T) \wedge s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = r_2\lceil(\mathbf{a}S \div \mathbf{a}T)$$
$$\wedge\ s_2\lceil(\mathbf{a}S \div \mathbf{a}T) = r_4\lceil(\mathbf{a}S \div \mathbf{a}T))$$

(End of Proof)

Lemma 5.17 : For connectable $\mathbf{C}_4$'s S and T, for traces s, s_0, t, and t_0, and for symbols $a \in \mathbf{i}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{o}(S\,\mathbf{b}\,T)$ such that $s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s$ and $t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t$

$$s_0 b a t_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge sabt \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow s_0 a b t_0 \in \mathbf{t}(S\,\mathbf{w}\,T)$$

Proof : By mathematical induction on the length of t_0.

Base : $t_0 = \epsilon$. Now we derive

$$s_0 b a t_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge sabt \in \mathbf{t}(S\,\mathbf{b}\,T)$$

$\Rightarrow$ { $\mathbf{t}(S\,\mathbf{w}\,T)$ and $\mathbf{t}(S\,\mathbf{b}\,T)$ are prefix-closed }

$$s_0 b \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge sa \in \mathbf{t}(S\,\mathbf{b}\,T)$$

$\Rightarrow$ { distribution of projection over concatenation, using $s = s_0\lceil(\mathbf{a}S \div \mathbf{a}T)$ and $a \in \mathbf{a}(S\,\mathbf{b}\,T)$ }

$$s_0 b \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 a\lceil(\mathbf{a}S \div \mathbf{a}T) \in \mathbf{t}(S\,\mathbf{b}\,T)$$

$\Rightarrow$ { Lemma 3.5, using $a \in \mathbf{i}(S\,\mathbf{b}\,T)$ and the definition of blending }

$$s_0 b \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 a \in \mathbf{t}(S\,\mathbf{w}\,T)$$

$\Rightarrow$ { Lemma 5.14, using $a \in \mathbf{i}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{o}(S\,\mathbf{b}\,T)$ }

$$s_0 ab \in \mathbf{t}(S\,\mathbf{w}\,T)$$

$=$ { $t_0 = \epsilon$ }

$$s_0 a b t_0 \in \mathbf{t}(S\,\mathbf{w}\,T)$$

Step : $t_0 = t_1 c$. We distinguish two cases : (i) $c \in \mathbf{a}S \cap \mathbf{a}T$ and (ii) $c \in \mathbf{a}(S\,\mathbf{b}\,T)$.

(i) $c \in \mathbf{a}S \cap \mathbf{a}T$. Hence,

$$t_1\lceil(\mathbf{a}S \div \mathbf{a}T) = t \tag{0}$$

Now we derive

$s_0bat_0 \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ sabt \in \mathbf{t}(S\ \mathbf{b}\ T)$
$= \ \{\ t_0 = t_1c,\ \mathbf{t}(S\ \mathbf{w}\ T)$ is prefix-closed $\}$
$s_0bat_1c \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ s_0bat_1 \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ sabt \in \mathbf{t}(S\ \mathbf{b}\ T)$
$\Rightarrow \ \{$ induction hypothesis, using (0) $\}$
$s_0bat_1c \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ s_0abt_1 \in \mathbf{t}(S\ \mathbf{w}\ T)$
$\Rightarrow \ \{$ Lemma 5.11 $\}$
$s_0abt_1c \in \mathbf{t}(S\ \mathbf{w}\ T)$
$= \ \{\ t_0 = t_1c\ \}$
$s_0abt_0 \in \mathbf{t}(S\ \mathbf{w}\ T)$

(ii) $c \in \mathbf{a}(S\ \mathbf{b}\ T)$. Since $t_0\lceil(\mathbf{a}\,S \div \mathbf{a}\,T) = t$ we may assume trace t_2 to be such that

$$t = t_2c \tag{1}$$

and, hence, since $t_0 = t_1c$

$$t_1\lceil(\mathbf{a}\,S \div \mathbf{a}\,T) = t_2 \tag{2}$$

Now we derive

$s_0bat_0 \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ sabt \in \mathbf{t}(S\ \mathbf{b}\ T)$
$\Rightarrow \ \{\ t_0 = t_1c$, (1), and $\mathbf{t}(S\ \mathbf{w}\ T)$ and $\mathbf{t}(S\ \mathbf{b}\ T)$ are prefix-closed $\}$
$s_0bat_1c \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ s_0bat_1 \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ sabt_2 \in \mathbf{t}(S\ \mathbf{b}\ T) \ \wedge\ sabt_2c \in \mathbf{t}(S\ \mathbf{b}\ T)$
$\Rightarrow \ \{$ induction hypothesis, using (2) $\}$
$s_0bat_1c \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ s_0abt_1 \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ sabt_2c \in \mathbf{t}(S\ \mathbf{b}\ T)$
$= \ \{\ s_0\lceil(\mathbf{a}\,S \div \mathbf{a}\,T) = s$ and $t_1\lceil(\mathbf{a}\,S \div \mathbf{a}\,T) = t_2$ according to (2). Distribution of projection over concatenation, using $a \in \mathbf{a}\,S \div \mathbf{a}\,T$, $b \in \mathbf{a}\,S \div \mathbf{a}\,T$, and $c \in \mathbf{a}\,S \div \mathbf{a}\,T$ $\}$
$s_0bat_1c \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ s_0abt_1 \in \mathbf{t}(S\ \mathbf{w}\ T) \ \wedge\ s_0abt_1c\lceil(\mathbf{a}\,S \div \mathbf{a}\,T) \in \mathbf{t}(S\ \mathbf{b}\ T)$
$\Rightarrow \ \{$ Lemma 3.5, using the definition of blending, if $c \in \mathbf{i}(S\ \mathbf{b}\ T)$. Lemma 5.10, using $a \in \mathbf{i}(S\ \mathrm{b}\ T)$ and $b \in \mathbf{o}(S\ \mathbf{b}\ T)$, if $c \in \mathbf{o}(S\ \mathbf{b}\ T)$ $\}$
$s_0abt_1c \in \mathbf{t}(S\ \mathbf{w}\ T)$
$= \ \{\ t_0 = t_1c\ \}$
$s_0abt_0 \in \mathbf{t}(S\ \mathbf{w}\ T)$

(End of Proof)

Lemma 5.18 : For connectable $\mathbf{C}_4$'s S and T, for traces s and t, and for symbols $a \in \mathbf{i}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{a}(S\,\mathbf{b}\,T)$

$$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sbt \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow$$
$$(\exists s_0, s_1 :: s_0 a \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 b s_1 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge$$
$$s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s \wedge s_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$$

Proof :

$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sbt \in \mathbf{t}(S\,\mathbf{b}\,T)$

= { definition of blending. $\mathbf{t}(S\,\mathbf{w}\,T)$ is prefix-closed }

$(\exists s_0, s_1 :: sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge s_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 b s_1 \in \mathbf{t}(S\,\mathbf{w}\,T)$
$\wedge\, s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s \wedge s_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$

= { calculus and distribution of projection over concatenation, using $a \in \mathbf{a}S \div \mathbf{a}T$ }

$(\exists s_0, s_1 :: s_0 a \lceil (\mathbf{a}S \div \mathbf{a}T) \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge s_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge$
$s_0 b s_1 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s \wedge s_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$

= { Lemma 3.5, since $a \in \mathbf{i}(S\,\mathbf{b}\,T)$, using the definition of blending }

$(\exists s_0, s_1 :: s_0 a \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 b s_1 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge$
$s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s \wedge s_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$

(End of Proof)

Example 5.0

Consider trace structure S with input alphabet $\{x, y\}$, output alphabet $\{a, b\}$, and command $x\,; a \mid y\,; b$ and consider trace structure T with output alphabet $\{x, y\}$, input alphabet $\varnothing$, and command $x \mid y$. Then S is, according to the rules, a $\mathbf{C}_2$ and T a $\mathbf{C}_3$. Alphabet $\{x, y\}$ is independent with respect to both trace structures. Moreover, $\mathbf{a}S \cap \mathbf{a}T = (\mathbf{o}S \cap \mathbf{i}T) \cup (\mathbf{o}T \cap \mathbf{i}S)$ and $S \lceil \{x, y\} = T \lceil \{x, y\}$, as a consequence of which S and T are connectable. The trace set of $S\,\mathbf{w}\,T$ is $\{\epsilon, x, y, xa, yb\}$ and the trace set of $S\,\mathbf{b}\,T$ equals $\{\epsilon, a, b\}$. Neither xb nor ya is an element of $\mathbf{t}(S\,\mathbf{w}\,T)$. Therefore, taking for s, t, u, a, and b in Lemma 5.16 ϵ, ϵ, ϵ, a, and b respectively, there do not exist traces s_0, s_1, and s_2 with the properties as in Lemma 5.16. Consequently, Lemma 5.16 does not hold when replacing $\mathbf{C}_2$ by $\mathbf{C}_3$ (or $\mathbf{C}_4$).

(End of Example)

5.4. The closure of $\mathbf{C}_1$

The blend of two connectable $\mathbf{C}_1$'s satisfies $\mathbf{R}_0$ through $\mathbf{R}_3$, as has been proved in Section 5.2. What remains are the proofs for $\mathbf{R}_4'$ and $\mathbf{R}_5'$. We prove $\mathbf{R}_4'$ for the composite of $\mathbf{C}_2$'s, which is sufficient since $\mathbf{C}_1 \subset \mathbf{C}_2$.

Lemma 5.19 : For connectable $\mathbf{C}_2$'s S and T, for traces s and t, and for symbols $a \in \mathbf{a}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{a}(S\,\mathbf{b}\,T)$ of different types

$$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sbat \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow sabt \in \mathbf{t}(S\,\mathbf{b}\,T)$$

Proof :

$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sbat \in \mathbf{t}(S\,\mathbf{b}\,T)$

$\Rightarrow$ { Lemma 5.16, symbols a and b are distinct since they are of different types }

$(\exists s_0, s_1, s_2 :: s_0as_1 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0bs_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s$
$\wedge\ s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = \epsilon \wedge s_2\lceil(\mathbf{a}S \div \mathbf{a}T) = at \wedge a \neq b)$

$\Rightarrow$ { $\mathbf{t}(S\,\mathbf{w}\,T)$ is prefix-closed. Renaming and calculus, using $a \in \mathbf{a}S \div \mathbf{a}T$ }

$(\exists s_0, s_1, s_2 :: s_0a \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0bs_1as_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s$
$\wedge\ s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = \epsilon \wedge s_2\lceil(\mathbf{a}S \div \mathbf{a}T) = t \wedge bs_1\lceil\{a\} = \epsilon)$

$\Rightarrow$ { if $a \in \mathbf{i}(S\,\mathbf{w}\,T)$ we apply Lemma 5.0 followed by Lemma 5.9. If not, then $a \in \mathbf{o}(S\,\mathbf{w}\,T)$ and we apply Lemma 5.6 }

$(\exists s_0, s_1, s_2 :: s_0abs_1s_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s\ \wedge$
$s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = \epsilon \wedge s_2\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { definition of blending, using $a \in \mathbf{a}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{a}(S\,\mathbf{b}\,T)$ }

$sabt \in \mathbf{t}(S\,\mathbf{b}\,T)$

(End of Proof)

Lemma 5.20 : For connectable $\mathbf{C}_1$'s S and T, for trace s, and for distinct symbols $a \in \mathbf{a}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{a}(S\,\mathbf{b}\,T)$

$$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sb \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow sab \in \mathbf{t}(S\,\mathbf{b}\,T)$$

Proof :

$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sb \in \mathbf{t}(S\,\mathbf{b}\,T)$

$\Rightarrow$ { Lemma 5.16 }

$$(\exists s_0, s_1, s_2 :: s_0 a s_1 \in \mathbf{t}(S \mathbf{w} T) \wedge s_0 b s_2 \in \mathbf{t}(S \mathbf{w} T) \wedge s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s$$
$$\wedge\ s_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = \epsilon \wedge s_2 \lceil (\mathbf{a}S \div \mathbf{a}T) = \epsilon)$$

$=$ { $\mathbf{t}(S \mathbf{w} T)$ is prefix-closed }

$$(\exists s_0 :: s_0 a \in \mathbf{t}(S \mathbf{w} T) \wedge s_0 b \in \mathbf{t}(S \mathbf{w} T) \wedge s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s)$$

$\Rightarrow$ { Lemma 5.12 }

$$(\exists s_0 :: s_0 ab \in \mathbf{t}(S \mathbf{w} T) \wedge s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s)$$

$\Rightarrow$ { definition of blending, using $a \in \mathbf{a}(S \mathbf{b} T)$ and $b \in \mathbf{a}(S \mathbf{b} T)$ }

$$sab \in \mathbf{t}(S \mathbf{b} T)$$

(End of Proof)

Now we have proved the following theorem.

Theorem 5.0 : $\mathbf{C}_1$ is closed under composition of connectable $\mathbf{C}_1$'s.

Example 5.1

Consider the C-wire element of Example 2.3 with input alphabet $\{a,q,r\}$ and output alphabet $\{b,p\}$ with command $a;(p;(q;b;a),r)^*$ and the And-element as introduced in Example 2.6 with input alphabet $\{p,d\}$, output alphabet $\{c,q,r\}$, and command $(p;c;d;q,r;p;(c;d;r),q)^*$. They are both $\mathbf{C}_1$'s. Alphabet $\{p,q,r\}$ is independent with respect to both trace structures, as has been argued in Examples 3.1 and 3.3. Each common symbol is input in the one and output in the other trace structure, and projected on $\{p,q,r\}$ both trace structures yield the trace structure with command $(p;q,r)^*$. As a consequence, they are connectable and their composite is specified by the blend, being $a;(c;d;b;a;(b;a),(c;d))^*$, which is a $\mathbf{C}_1$, indeed. This element may be interpreted as a Quick Return Linkage (QRL) [10]. It has a cyclic way of operation. In the first half of the cycle a component informs another component via a of the presence of input data, and is notified via b that these data have been processed. The other component is notified of these data via c and informs the first component via d that these data have been processed. The second half of the cycle, the return-to-zero phase, then proceeds without any communications between both components.

(End of Example)

Example 5.2

Consider the C-element with two outputs of Example 2.2 with input alphabet

$\{c,s\}$, output alphabet $\{d,t\}$, and command $c,s;((d;c),(t;s))^*$. Alphabets $\{c,d\}$ and $\{s,t\}$ are independent with respect to this element as has been argued in Example 3.0. This element may be composed with two QRL's of the preceding example. The first QRL is exactly the one derived in that example. The other one is initialized in a different state and its symbols are renamed. Its input alphabet is $\{p,t\}$, its output alphabet $\{r,s\}$, and its command is $(s;t;r;p;(r;p),(s;t))^*$. Alphabet $\{c,d\}$ is independent with respect to the first QRL, alphabet $\{s,t\}$ to the other QRL. The projections of the first QRL and of the C-element on $\{c,d\}$ yield the trace structure with command $(c;d)^*$. Since the input-in-the-one-and-output-in-the-other-one rule is obviously satisfied, these two components are connectable. Alphabet $\{s,t\}$ turns out to be independent with respect to the composite and the projections of the composite and the other QRL on $\{s,t\}$ yields the trace structure with command $(s;t)^*$. That makes these two components connectable as well. The result of their blending is $a;((b;a;b;a),(r;p;r;p))^*$. This may be interpreted as a binary semaphore [1]. Such a semaphore may be composed with another one, using $\{p,r\}$ for the one and $\{a,b\}$ for the other one as independent alphabet by means of which they are connected. The result is a ternary semaphore. In this way we can compose $k-1$ binary semaphores, which yields a k-ary semaphore.

(End of Example)

5.5. The closure of C_2

According to Section 5.2 and Lemma 5.19 the only rule left to prove is $\mathbf{R}_5''$.

Lemma 5.21 : For connectable $\mathbf{C}_2$'s S and T, for trace s, and for distinct symbols $a \in \mathbf{a}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{a}(S\,\mathbf{b}\,T)$, not both input symbols,

$$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sb \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow sab \in \mathbf{t}(S\,\mathbf{b}\,T)$$

Proof :

$$
\begin{aligned}
& sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sb \in \mathbf{t}(S\,\mathbf{b}\,T) \\
\Rightarrow\ & \{\ \text{Lemma 5.16}\ \} \\
& (\exists s_0, s_1, s_2 :: s_0 a s_1 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 b s_2 \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \\
& \qquad \wedge s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = \epsilon \wedge s_2\lceil(\mathbf{a}S \div \mathbf{a}T) = \epsilon) \\
=\ & \{\ \mathbf{t}(S\,\mathbf{w}\,T) \text{ is prefix-closed}\ \} \\
& (\exists s_0 :: s_0 a \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0 b \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s) \\
\Rightarrow\ & \{\ \text{Lemma 5.13}\ \} \\
& (\exists s_0 :: s_0 ab \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s)
\end{aligned}
$$

$\Rightarrow$ {definition of blending, using $a \in \mathbf{a}(S\ \mathbf{b}\ T)$ and $b \in \mathbf{a}(S\ \mathbf{b}\ T)$ }
$sab \in \mathbf{t}(S\ \mathbf{b}\ T)$

(End of Proof)

This means that we have proved the following theorem.

Theorem 5.1 : $\mathbf{C}_2$ is closed under composition of connectable $\mathbf{C}_2$'s.

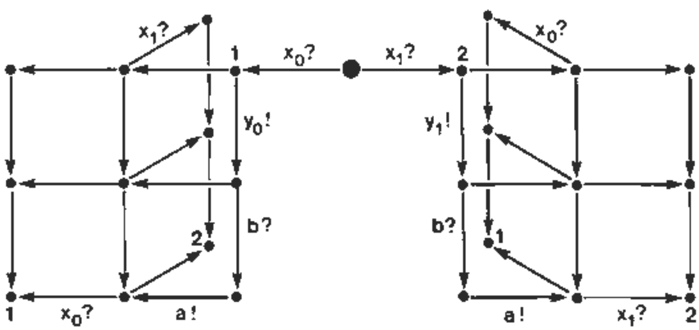
Example 5.3

Consider the Three-wire component of Example 2.4 with input alphabet $\{x_0, x_1, b\}$, output alphabet $\{y_0, y_1, a\}$, and command $(x_0 ; y_0 ; b ; a \mid x_1 ; y_1 ; b ; a)^*$. We can 'lengthen' these wires by composing this element with, apart from renaming, the same element. The latter is the component with input alphabet $\{y_0, y_1, c\}$, output alphabet $\{z_0, z_1, b\}$, and command $(y_0 ; z_0 ; c ; b \mid y_1 ; z_1 ; c ; b)^*$. Alphabet $\{y_0, y_1, b\}$ is independent with respect to both components as has been argued in Example 3.2, and the projections on $\{y_0, y_1, b\}$ yields for both trace structures the trace structure with command $((y_0 \mid y_1) ; b)^*$. The blend of the two, being the specification of the composite, is $(x_0 ; z_0 ; c ; a \mid x_1 ; z_1 ; c ; a)^*$. This is, apart from renaming, the same component as the ones that we started from.

(End of Example)

Example 5.4

Consider the buffer as introduced in Example 2.8 with input alphabet $\{x_0, x_1, b\}$, output alphabet $\{y_0, y_1, a\}$, and state graph



Alphabets $\{x_0, x_1, a\}$ and $\{y_0, y_1, b\}$ are independent. This buffer can be composed with another buffer that is obtained from this one by replacing every symbol by its alphabetical successor. The projections of both buffers onto the set of common symbols, i.e. $\{y_0, y_1, b\}$ are the trace structure with command $((y_0 \mid y_1); b)^*$. The other requirements for connectability are satisfied as well and, hence, we may compose these two buffers. A command for the specification of the composite, which is, of course, a two-place buffer, is hard to derive from these two specifications. In fact, any command for the composite is monstrous. Although it is clearly necessary to be able to reason about such a simple component in an adequate way, we consider it outside the scope of this monograph. Apparently, this is not the appropriate level of abstraction for deriving the specification of a composite. This is, as pointed out in the next chapter, one of the topics of future research.

(End of Example)

5.6. The closure of $\mathbf{C}_4$

Left to prove for the blend of two $\mathbf{C}_4$'s are $\mathbf{R}_4''$ and $\mathbf{R}_5'''$.

Lemma 5.22 : For connectable $\mathbf{C}_4$'s S and T, for traces s and t, and for symbols $a \in \mathbf{a}(S \mathbf{b} T)$, $b \in \mathbf{a}(S \mathbf{b} T)$, and $c \in \mathbf{a}(S \mathbf{b} T)$ such that b is of another type than a and c

$$sabtc \in \mathbf{t}(S \mathbf{b} T) \wedge sbat \in \mathbf{t}(S \mathbf{b} T) \Rightarrow sbatc \in \mathbf{t}(S \mathbf{b} T)$$

Proof : We distinguish two cases : (i) $c \in \mathbf{i}(S \mathbf{b} T)$ and (ii) $c \in \mathbf{o}(S \mathbf{b} T)$

(i) $c \in \mathbf{i}(S \mathbf{b} T)$. Now we derive

$sabtc \in \mathbf{t}(S \mathbf{b} T) \wedge sbat \in \mathbf{t}(S \mathbf{b} T)$

$=$ { definition of blending, using $a \in \mathbf{a}(S \mathbf{b} T)$ and $b \in \mathbf{a}(S \mathbf{b} T)$ }

$(\exists s_0, s_1, t_0 :: sabtc \in \mathbf{t}(S \mathbf{b} T) \wedge s_0 b s_1 a t_0 \in \mathbf{t}(S \mathbf{w} T) \wedge s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s$
$\wedge\ s_1 \lceil (\mathbf{a}S \div \mathbf{a}T) = \epsilon \wedge t_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { Lemma 5.0, since the type of a, being the type of c, is input. Moreover, $\mathbf{t}(S \mathbf{b} T)$ is prefix-closed and renaming }

$(\exists s_0, t_0 :: sabtc \in \mathbf{t}(S \mathbf{b} T) \wedge sabt \in \mathbf{t}(S \mathbf{b} T) \wedge s_0 bat_0 \in \mathbf{t}(S \mathbf{w} T) \wedge$
$s_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = s \wedge t_0 \lceil (\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { Lemma 5.17, since a is input and b is of another type than a }

$(\exists s_0, t_0 :: sabtc \in \mathbf{t}(S \mathbf{b} T) \wedge s_0 abt_0 \in \mathbf{t}(S \mathbf{w} T) \wedge s_0 bat_0 \in \mathbf{t}(S \mathbf{w} T) \wedge$

$s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \ \wedge\ t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$=$ { calculus and distribution of projection over concatenation, using $a \in \mathbf{a}(S\,\mathbf{b}\,T)$, $b \in \mathbf{a}(S\,\mathbf{b}\,T)$, and $c \in \mathbf{a}(S\,\mathbf{b}\,T)$ }

$(\exists s_0, t_0 :: s_0abt_0c\lceil(\mathbf{a}S \div \mathbf{a}T) \in \mathbf{t}(S\,\mathbf{b}\,T) \ \wedge\ s_0abt_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge$
$s_0bat_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge\ s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \ \wedge\ t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { Lemma 3.5, using $c \in \mathbf{i}(S\,\mathbf{b}\,T)$ and the definition of blending }

$(\exists s_0, t_0 :: s_0abt_0c \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge\ s_0bat_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge$
$s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \ \wedge\ t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { Lemma 5.10 }

$(\exists s_0, t_0 :: s_0bat_0c \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge\ s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \ \wedge\ t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { definition of blending, using $a \in \mathbf{a}(S\,\mathbf{b}\,T)$, $b \in \mathbf{a}(S\,\mathbf{b}\,T)$, and $c \in \mathbf{a}(S\,\mathbf{b}\,T)$ }

$sbatc \in \mathbf{t}(S\,\mathbf{b}\,T)$

(ii) $c \in \mathbf{o}(S\,\mathbf{b}\,T)$. In this case we derive

$sabtc \in \mathbf{t}(S\,\mathbf{b}\,T) \ \wedge\ sbat \in \mathbf{t}(S\,\mathbf{b}\,T)$

$=$ { definition of blending }

$(\exists s_0, s_1, t_0 :: s_0as_1bt_0c \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge\ sbat \in \mathbf{t}(S\,\mathbf{b}\,T) \ \wedge\ s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s$
$\wedge\ s_1\lceil(\mathbf{a}S \div \mathbf{a}T) = \epsilon \ \wedge\ t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { Lemma 5.1, since the type of a, being the type of c, is output. Moreover, $\mathbf{t}(S\,\mathbf{w}\,T)$ is prefix-closed and renaming }

$(\exists s_0, t_0 :: s_0abt_0c \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge\ s_0abt_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge\ sbat \in \mathbf{t}(S\,\mathbf{b}\,T)$
$\wedge\ s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \ \wedge\ t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { Lemma 5.17, since a is output and b of another type than a }

$(\exists s_0, t_0 :: s_0abt_0c \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge\ s_0bat_0 \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge$
$s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \ \wedge\ t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { Lemma 5.10 }

$(\exists s_0, t_0 :: s_0bat_0c \in \mathbf{t}(S\,\mathbf{w}\,T) \ \wedge\ s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s \ \wedge\ t_0\lceil(\mathbf{a}S \div \mathbf{a}T) = t)$

$\Rightarrow$ { definition of blending, using $a \in \mathbf{a}(S\,\mathbf{b}\,T)$, $b \in \mathbf{a}(S\,\mathbf{b}\,T)$, and $c \in \mathbf{a}(S\,\mathbf{b}\,T)$ }

$sbatc \in \mathbf{t}(S\,\mathbf{b}\,T)$

(End of Proof)

Lemma 5.23 : For connectable C_4's S and T, for trace s, and for symbols $a \in \mathbf{a}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{a}(S\,\mathbf{b}\,T)$ of different types

$$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sb \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow sab \in \mathbf{t}(S\,\mathbf{b}\,T)$$

Proof : We assume $a \in \mathbf{i}(S\,\mathbf{b}\,T)$ and, hence, have to prove

$$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sb \in \mathbf{t}(S\,\mathbf{b}\,T) \Rightarrow sab \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sba \in \mathbf{t}(S\,\mathbf{b}\,T)$$

$sa \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sb \in \mathbf{t}(S\,\mathbf{b}\,T)$

$=$ { Lemma 5.18, using that $\mathbf{t}(S\,\mathbf{w}\,T)$ is prefix-closed }

$(\exists s_0 :: s_0a \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0b \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s)$

$\Rightarrow$ { Lemma 5.14 }

$(\exists s_0 :: s_0ab \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0ba \in \mathbf{t}(S\,\mathbf{w}\,T) \wedge s_0\lceil(\mathbf{a}S \div \mathbf{a}T) = s)$

$\Rightarrow$ { definition of blending, using $a \in \mathbf{a}(S\,\mathbf{b}\,T)$ and $b \in \mathbf{a}(S\,\mathbf{b}\,T)$ }

$sab \in \mathbf{t}(S\,\mathbf{b}\,T) \wedge sba \in \mathbf{t}(S\,\mathbf{b}\,T)$

(End of Proof)

This completes the proof of

Theorem 5.2 : C_4 is closed under composition of connectable C_4's.

Example 5.5

Consider the C_3 of Example 2.12 with input alphabet $\{a, d, e\}$, output alphabet $\{b, c, f\}$, and command $(((f\,;a),(b\,;d))^*\,;f\,;a\,;\ (c\,;e\,;b\,;d)^*\,;b\,;d)^*$. As argued in Example 3.6, alphabet $\{c, e\}$ is independent. Projection on $\{c, e\}$ yields the trace structure with command $(c\,;e)^*$. The trace structure with command $(c\,;e)^*$, input alphabet $\{c\}$, and output alphabet $\{e\}$ is a C_1. These two components are connectable. Composition of the two yields the projection of the first trace structure onto $\{a, b, d, f\}$, which is, according to Example 3.6, not a C_3.

(End of Example)

6

Suggestions for further study

The theory developed in this monograph provides a base for a theory on delay-insensitive circuits. In this chapter we point out a number of generalizations that might be considered.

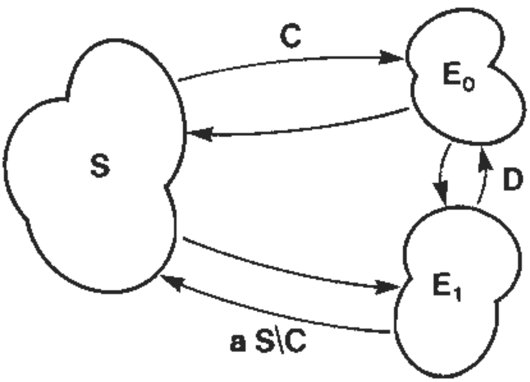
In Chapter 3 we noticed already that the requirements for connectability of trace structures S and T are rather restrictive. The first relaxation considered relates to the requirement that their projections on the set of common symbols be the same. This is, provided that the set of common symbols is independent with respect to both trace structures, a sufficient condition to guarantee absence of computation interference. There is nothing wrong, however, with a situation in which the one component is able to receive an input that the other component is never able to produce as output. An example of this kind is a variable, as we have introduced in Example 2.7, that is composed with another component that always retrieves a stored value twice before storing a next value in the variable. It might be sufficient to require

$$(\forall a : a \in \mathbf{o}S \cap \mathbf{i}T : sa \in \mathbf{t}S\lceil(\mathbf{a}S \cap \mathbf{a}T) \Rightarrow sa \in \mathbf{t}T\lceil(\mathbf{a}S \cap \mathbf{a}T)) \wedge$$
$$(\forall a : a \in \mathbf{o}T \cap \mathbf{i}S : sa \in \mathbf{t}T\lceil(\mathbf{a}S \cap \mathbf{a}T) \Rightarrow sa \in \mathbf{t}S\lceil(\mathbf{a}S \cap \mathbf{a}T))$$

for all traces $s \in \mathbf{t}S\lceil(\mathbf{a}S \cap \mathbf{a}T) \cap \mathbf{t}T\lceil(\mathbf{a}S \cap \mathbf{a}T)$. When taking delays into account it is not obvious that this requirement is sufficient to guarantee absence of computation interference. This should be proved again by means of composability of traces as we did in Chapter 4 for the more restrictive composition operator. Expressing requirements in terms of individual traces is undesirable, however. We would like to express this or a more suitable requirement in terms of trace structures. How this should be done remains to be seen.

Connecting trace structures by means of independent alphabets seems too restrictive a requirement as well. Connecting two wires with one another in the usual way, or connecting a C-element with a Fork to obtain a C-element with two outputs is still impossible. We would like to be able to compose trace structures with a set of common symbols that is not independent with respect to each of them. How to incorporate this kind of composition is not clear yet. The following might be a possible strategy.

Consider two components that are specified by trace structures S and T respectively. Let C be the set of common symbols and let C be independent with respect to neither S nor T. For trace structure S this means that there is a trace $t \in \mathbf{t}S$ and a symbol $a \in C \cap \mathbf{i}S$ such that $(t \lceil C)a \in \mathbf{t}S \lceil C$ and $ta \notin \mathbf{t}S$ (or something similar with C replaced by $\mathbf{a}S \setminus C$). In other words, there are still communications to be performed by means of the symbols of the complement with respect to $\mathbf{a}S$ before input a can be received by the component. The two environments that the environment of S is partitioned into by C and $\mathbf{a}S \setminus C$ cannot communicate with the component independent of one another. They need additional information on each other's progress. Therefor we could introduce an alphabet D of fresh symbols via which the two environments can directly communicate. In order to reflect these communications traces of S should be interspersed with symbols of D in a suitable way. The component-environment pair now becomes a triple :



Having done something similar with trace structure T, using the same set of symbols D, we can blend the new S and T, provided that $S \lceil (C \cup D)$ and $T \lceil (C \cup D)$ meet certain conditions, e.g. $S \lceil (C \cup D) = T \lceil (C \cup D)$.

The problem of course is the interspersion of traces of S with symbols of D. One of the questions is what requirements to impose upon the resulting trace structure. A necessary, and possibly sufficient, condition seems to be that the projections of the new trace structure onto $C \cup D$ and $(\mathbf{a}S \setminus C) \cup D$ be delay-insensitive. A second question is how to find D and how to construct the desired trace structure. A trivial way is to conceive one of the environments, E_0 say, as a pass-through for all incoming signals. This means that there is a one-to-one correspondence between input symbols of C and output symbols of D and between output symbols of C and input symbols of D. (Input and output is here with respect to E_0.) Moreover, in the specification of the communications via C and D every input is followed by its corresponding output and so repeatedly. The specification of the communications via D and $\mathbf{a}S \setminus C$ is in this case the same as the one via C and $\mathbf{a}S \setminus C$ with every symbol of C replaced by its corresponding symbol in D.

Once we have properly relaxed the requirements for connectability and have proved the absence of transmission and computation interference we have to answer the question whether and how to incorporate multiple transitions on a wire in our formalism. As we have pointed out in the introduction, we can allow multiple transitions on a wire in the presence of a data valid wire that signals the validity of a voltage level on the first wire. This kind of protocol is often used for data transmission. A high level on a wire represents a logical one and a low level on that wire a logical zero. Having n data wires we can convey 2^n different values.

Not using a data valid wire we encode data by a so-called m out of n coding. Having n wires a transition on exactly m of them, $0 \leqslant m \leqslant n$, represents a value. In this way we can convey $\binom{n}{m}$ different values. For fixed n the maximum value of $\binom{n}{m}$ is asymptotically $2^n / \sqrt{n}$. Notice that we have used a 1 out of 2 coding for the data transmission in the examples.

The advantage of data transmission with a data valid wire is the smaller number of wires needed and the availability of circuits that can handle data encoded in this way, e.g. adders and multipliers. The number of wires used to convey data, however, is typically 8, which makes, together with the data valid wire, a total of 9 wires required. An m out of n coding requires 11 wires for 8 bits of information,which does not seem to be too large a difference. An interesting question is how to build arithmatical circuits that can handle data encoded in this way. It might just be that this encoding seems more difficult only because we are used to the other one.

A question, which is often posed, is whether there exists a (finite) base for delay-insensitive circuits, i.e. a (finite) set of delay-insensitive circuits by means of which we can obtain all delay-insensitive circuits by composition. Once we have relaxed the requirements for composition of components this is a valid question. It might be that there exists a base consisting of just a few elements, which would make a gate array approach for the implementation of a component as chip very attractive. Closure properties of classes may be helpful in finding such a set. Using, for instance, the composition operator as defined in Chapter 3 we cannot obtain a $\mathbf{C}_4$ from $\mathbf{C}_2$'s. This means that an arbiterlike device necessarily belongs to a base. It is very likely that the closure properties derived in Chapter 5 hold for less restrictive composition operators as well.

Trace theory as it is used here provides the first step towards a high level specification language that we would like a silicon compiler, our ultimate goal, to be able to accept. Specifying circuits at the current level of abstraction is a nuisance. Another topic of research, therefore, is how to translate specifications written at a higher level of abstraction, like for instance the specifications in [12], into specifications that satisfy the rules for delay-insensitivity and still have, in some sense, the same meaning. One can think, for example, of adding symbols to guarantee proper communications.

7

Concluding remarks

In this monograph we have discussed specifications of circuits when making no assumptions on wire delays. This has led to a definition and a classification of delay-insensitive circuits. Moreover, we have proposed a composition operator that we have shown to warrant internal communications that are free of transmission and computation interference. Three of the four classes turn out to be closed under this composition operator. A few final remarks on the results obtained seem to be apposite.

C_1, C_2, and C_3 arose from an intuitive understanding of delay-insensitive circuits and of decisions that are to be made in the component and in the environment. Since C_3 turned out not to be closed under the composition operator proposed, the need for a larger, still physically interpretable, class developed. C_4 is a class that satisfies these requirements, which makes C_3, in fact, obsolete.

Petri nets [7] are frequently used for the specification of delay-insensitive circuits. They suffer, however, from a canonical form problem, i.e. distinct Petri nets may specify the same circuit. This makes it hard to capture properties of delay-insensitive circuits in terms of Petri nets. Trace structures do not suffer from this canonical form problem and are, therefore, more suited to define and classify delay-insensitive components. Petri nets can, like our program texts or state graphs, very well be used for the representation of trace structures. The question whether there is a representation that should be preferred to the others is not easily answered. Probably it depends on the circumstances under which they are to be used and on the question by whom they are to be used.

We have confined our attention to components that satisfy the rules for delay-insensitivity and we have defined for that class of components a composition operator. The advantage of this approach is that it is not necessary to take wire delays into account when composing components : the blend has been shown in Chapter 4, with some effort but we only have to do it once, to be a proper composition operator for this kind of components. This is opposed to the approach taken in [12] where a larger class of components is considered. When composing this kind of components, however, one cannot simply use the blend but one needs a much more complicated composition operator, called agglutination. The result of such an agglutination is not easily computed. Confining the class of components to be considered seems to be a better approach for dealing with wire delays.

References

[0] T.J. Chaney, C.E. Molnar, Anomalous Behavior of Synchronizer and Arbiter Circuits, IEEE Transactions on Computers, Vol C-22, 1973, pp 421-422.

[1] Edsger W. Dijkstra, Cooperating Sequential Processes, in Programming Languages (F. Genuys ed.), Academic Press, 1968, pp 43-112.

[2] Edsger W. Dijkstra, Lecture Notes 'Predicate Transformers' (Draft), EWD 835, 1982.

[3] T.P. Fang, C.E. Molnar, Synthesis of Reliable Speed-independent Circuit Modules, Part 1 and 2, Technical Memoranda No. 297 and 298, Computer Systems Laboratory, Institute for Biomedical Computing, Washington University, St. Louis, Missouri, 1983.

[4] L.R. Marino, General Theory of Metastable Operation, IEEE Transactions on Computers, Vol C-30, No. 2, 1981, pp 107-115.

[5] C. Mead, L. Conway, Introduction to VLSI Systems, Addison-Wesley, 1980.

[6] Raymond E. Miller, Switching Theory, Wiley, 1965, Vol. 2, Chapter 10.

[7] J.L. Peterson, Petri nets, Computing Surveys, Vol. 9, No. 3, 1977.

[8] Science and the citizen, Scientific American, Vol. 228, 1973, pp 43-44.

[9] C.L. Seitz, Self-timed VLSI Systems, Proceedings of the Caltech Conference on VLSI, 1979, pp 345-355.

[10] C.L. Seitz, Private Communication.

[11] C.L. Seitz, System Timing, in [5], pp 218-262.

[12] Jan L.A. van de Snepscheut, Trace Theory and VLSI Design, Ph. D. Thesis, Department of Computing Science, Eindhoven University of Technology, 1983.

[13] I.E. Sutherland, C.E. Molnar, R.F. Sproull, J.C. Mudge, The Trimosbus, Proceedings of the Caltech Conference on VLSI, 1979, pp 395-427.

Subject index

Samenvatting

In dit proefschrift wordt een definitie en een classificatie van en een compositiemethode voor vertragingsongevoelige circuits besproken. Dit zijn circuits waarvoor geen aannamen gemaakt worden omtrent vertragingen in verbindingsdraden of omtrent de snelheid waarmee zo een circuit reageert op input signalen. De reden voor de bestudering van dergelijke circuits is tweeerlei. Enerzijds bestaan er circuits die niet altijd binnen een bepaalde tijd een berekening hebben uitgevoerd. Dit betekent dat er in de specificatie van een circuit dat met zo een circuit wordt verbonden niet van uitgegaan mag worden dat input signalen binnen een zekere tijd na de output signalen zullen kunnen worden ontvangen. Anderzijds blijkt dat door het verkleinen van geintegreerde schakelingen de vertragingstijden van elektrische signalen in verbindingsdraden toenemen vergeleken met de schakeltijden van transistoren, zodat vertragingen in draden niet langer verwaarloosd mogen worden.

Een viertal klassen van vertragingsongevoelige circuits wordt op axiomatische wijze gedefinieerd. Drie van deze klassen blijken gesloten te zijn onder de voorgestelde compositieoperator, terwijl de vierde dit niet is. Voor de specificatie en compositie van circuits en voor de geslotenheidsstellingen wordt gebruik gemaakt van trace theory. Dit is een theorie van symboolrijen en verzamelingen symboolrijen.

Bij het samenstellen van circuits dient aan twee voorwaarden te zijn voldaan. Ten eerste moet gegarandeerd zijn dat elektrische signalen op een verbindingsdraad niet met elkaar kunnen interfereren. Door geen aannamen te maken over vertragingen betekent dit dat hooguit één signaal per draad is toegestaan. Daarnaast mag een elektrisch signaal pas bij een circuit arriveren als dat circuit, volgens zijn specificatie, in staat is tot de ontvangst van dat signaal. Van de in dit proefschrift voorgestelde compositieoperator wordt aangetoond dat bij compositie van vertragingsongevoelige circuits aan deze beide voorwaarden is voldaan.

Curriculum vitae

De schrijver van dit proefschrift is op 11 juni 1953 geboren te Den Helder. Na het eindexamen Gymnasium-β in 1971 te hebben afgelegd aan het Drachtster Lyceum te Drachten is een aanvang gemaakt met de studie wiskunde aan de Technische Hogeschool Eindhoven. In februari 1980 wordt het diploma wiskundig ingenieur behaald, na afstudeerwerk onder leiding van prof.dr. N.G. de Bruijn. Tot april 1982 wordt daarna gewerkt als medewerker van de Sector Informatica aan het Dr. Neher Laboratorium van de PTT in Leidschendam. Sinds 15 april 1982 wordt als wetenschappelijk medewerker aan de Onderafdeling der Wiskunde en Informatica van de Technische Hogeschool Eindhoven gewerkt in de vakgroep Informatica onder leiding van prof.dr. M. Rem. Van september tot en met november 1983 is bovendien als research fellow onderzoek verricht op het gebied van vertragingsongevoelige systemen onder leiding van prof.dr. C.E. Molnar aan de Washington University te St. Louis, Missouri.

STELLINGEN

behorende bij het proefschrift

Classification and composition of
delay-insensitive circuits

van

Jan Tijmen Udding

Eindhoven,
25 september 1984

0. Voor iedere natuurlijke q en m waarvoor geldt $m \neq 1$, $q \geqslant 2m+1$, $q \equiv (1$ of $3 \mathbf{mod} 6)$ en $m \equiv (1$ of $3 \mathbf{mod} 6)$ bestaan er twee Steiner triple systems van orde q (op dezelfde verzameling punten) die precies een Steiner triple system van orde m gemeen hebben.

   lit : J.I. Hall and J.T. Udding, On the Intersection of Pairs of Steiner Triple Systems, Proc. Kon. Akad. v. Wet., A80, 1977, pp 87-100.

1. Voor gegeven alfabet A en prefix-closed trace set U heeft de vergelijking $T \subseteq A^{*} : T = U \mathbf{b}\, s.T$ precies één oplossing die ϵ bevat indien voor iedere $u \in U$ geldt $\mathbf{l}(u \lceil s.A) \leqslant \mathbf{l}(u \lceil A)$.

   lit : J.T. Udding, On recursively defined sets of traces, Intern Memorandum, JTU0a, 1983.

2. De klassen van vertragingsongevoelige trace structures die voldoen aan de regels $\mathbf{R}_0$ tot en met $\mathbf{R}_4''$ en aan ofwel $\mathbf{R}_5'$ dan wel $\mathbf{R}_5''$ zijn gesloten onder compositie van connectable trace structures.

   lit : Dit proefschrift.

3. Het is opvallend en valt te betreuren dat in zo weinig boeken over tralietheorie aandacht wordt besteed aan eigenschappen van morfismen.

4. Het blijven uitbreiden van het relationele model draagt geenszins bij tot een goede fundering van de theorie over informatiesystemen.

5. Imperatieve programmeertalen zijn een erfenis uit de tijd dat het doel van een taal nog was het programmeren van een machine. Nu machines er zijn om onze programma's uit te voeren, dient aanmerkelijk meer aandacht te worden besteed aan het gebruik van non-imperatieve programmeertalen dan momenteel het geval is.

6. Bij het beschrijven van fysische objecten door middel van een wiskundig model dienen objecten die in het model van elkaar verschillen te corresponderen met objecten die om fysische redenen van elkaar verschillen. Petri netten dienen derhalve niet gebruikt te worden voor de specificatie van vertragingsongevoelige systemen.

7. Slechte ervaringen met inadequate formalismen hebben geleid tot een schromelijke onderschatting van de mogelijke rol van formalismen.

8. Het idee dat bewijsvoering gereduceerd kan worden tot formulemanipulatie getuigt van een schromelijke overschatting van de mogelijke rol van formalismen.

9. Onderzoek heeft bij uitstek een individueel karakter. De te ver doorgevoerde demokratisering van het universitair bestel in Nederland is dan ook funest voor het verrichten van goed en origineel onderzoek.

10. Gelukkig is, zoals de naam al zegt, temporele logica maar tijdelijk.

11. Binnenkort zal de zogenaamde 'school met de computer' zijn intrede doen in de strijd om de gunst van de leerplichtige. De suggestie als zou een dergelijke school een streepje voor hebben op andere scholen is onjuist en dient als misleiding te worden aangemerkt.