Networks of communicating processes and their (de-)composition

Citation for published version (APA):

Chen, W., Udding, J. T., & Verhoeff, T. (1989). *Networks of communicating processes and their (de-)composition*. (Computing science notes; Vol. 8905). Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/1989

Document Version:

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

Please check the document version of this publication:

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

[Link to publication](#)

General rights

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

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

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

**Networks of Communicating Processes
and Their (De-)Composition**

by

**Wei Chen, Jan Tijmen Udding,
Tom Verhoeff**

89/5

May, 1989

COMPUTING SCIENCE NOTES

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

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
All rights reserved
editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.

Networks of Communicating Processes and Their (De-)Composition

Wei Chen

Department of Computer Science
Washington University
St. Louis, MO

Jan Tijmen Udding

Department of Mathematics and Computing Science
Groningen University
Groningen, The Netherlands

Tom Verhoeff

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

March 1989

Abstract

In this paper we present a general framework within which a study of networks of processes can be conducted. It is based upon the mathematical technique to abstract from irrelevant detail. We start out with a large class of objects and some operations upon them. Depending upon a correctness criterion to be imposed, some of these objects turn out to be equivalent. The resulting space of equivalence classes and operations upon them is, under certain conditions, the (fully) abstract space of interest for that particular correctness concern.

We use this approach to study networks for which we assume the communications to be asymmetric and asynchronous. We impose the correctness criterion of absence of computation interference. The resulting abstract space turns out to be the space of delay-insensitive specifications. As operator we study composition of networks. The composition operator on the resulting space is shown to have a surprisingly simple factorization property, the prove of which turns out to be very simple due to the approach taken.

0 Introduction

In this paper, which is an extended version of [0], we study networks of communicating processes. The communications between processes are asymmetric, that is, for each communication there is an initiating party which sends a signal to other parties regardless of their readiness. We study both synchronous (all parties complete their parts of the communication at the same step) and asynchronous (each party is free to complete its part of the communication at its own convenience) communications. The overall goal is to model computations implemented as electrical circuits—where communications are typically asynchronous—in a mathematical way, so we can derive programs to be implemented as circuits in the same way as we derive computer programs. Since most concurrent programming languages are based upon synchronous symmetric communication primitives (and rightfully so!), it is important to establish a semantically consistent link between synchronous and asynchronous based communication primitives.

The approach we take to this problem is quite general and widely applicable when modeling the operation of systems. We start out from a large class of primitive objects. The kind of the objects we consider is discussed in the next section. Some of these objects are just as “good” as other objects for implementing a particular specification. A well-known example of such an ordering of objects is the one induced by non-determinism. Removing non-determinism yields a program which is at least as good as the original one. In general, a certain correctness concern induces an ordering on the objects under consideration. We call two objects equivalent if either of the two is as good as the other one. This induces equivalence classes of objects that cannot be distinguished under that particular correctness concern. In this way each correctness concern induces a (fully abstract) space of equivalence classes. We illustrate this approach by choosing absence of computation interference as our correctness concern. Absence of computation interference means that no signal will arrive at a process when that process is not ready for it according to its specification.

For this particular correctness concern it turns out that the induced equivalence classes yield an obvious way to define a denotational semantics of parallel composition. It is shown to have properties such as monotonicity, associativity, and continuity. More importantly, we show how composition can be used to “mechanically” decompose a given specification into smaller ones, the composition of which satisfies the original specification. Roughly speaking, if the composition of an unknown component X and a given component M is to satisfy a certain specification S , then it is good enough too look for X ’s satisfying the composition of S and the “inverse” of M . This is very similar to the situation in sequential programming where, given an

invariant of a repetition and a statement establishing a step towards termination, the specification of the rest of the body of the repetition can be computed.

The authors believe that similar, but more powerful, algebraic laws will hold when other correctness concerns, such as progress, are taken into consideration.

Overview

In Section 1 we describe the formal model. It introduces a space of processes and a space of process networks together with a network composition operator. For the closed networks we define two operational semantics—one for synchronous and one for asynchronous communication—and a related correctness concern. This induces two natural satisfaction relations on the space of networks. To each satisfaction relation corresponds a notion of equivalence for networks.

Section 2 studies the space of networks under the equivalence relation based on asynchronous communication and its correctness concern. The major result is that each network is equivalent to a network consisting of single process. This result is used in Section 3 to give a fully abstract denotational semantics for the composition of processes. We show that composition has certain interesting properties. We also present a factorization theorem that can be used for decomposition.

Finally, in Section 4 we comment on our approach and briefly compare it to that of others.

1 Networks of communicating processes

In this section we define our primitive objects and the operations on them, and we give a mechanistic appreciation so as to provide the reader with some intuition behind the definitions. We emphasize that this intuition is unnecessary, and can even be dangerous, once the formal basis of our objects and operations has been given. We only want to sustain the claim that the purely mathematical game that we are playing may have some practical significance.

1.0 Structure of systems

The objects we wish to study are networks of communicating processes, also called systems. We are only interested in the external behavior of a process. Externally, a process in a network communicates with another process by sending signals to and receiving signals from that process via so-called *links*.

The set of links via which processes can communicate with one another is called Σ . Each sending and each receiving of a signal constitutes an event. These events are atomic actions, that is, there is no notion of overlapping events. The event “sending a signal” onto a certain link is denoted by the name of the link paired with an exclamation mark. The event “receiving a signal” on a certain link is denoted by the name of the link paired with a query mark.

We characterize a process by the set of events it can engage in, called its alphabet, and in what order those events may occur, which is captured by its trace set. We formalize these concepts in the following way.

Definition 0 A *directed* symbol is an element of $\Omega = \Sigma \times \{?, !\}$. A directed symbol is called an input symbol when its second component is ‘?’ and an output symbol otherwise. $\square$

We have called the elements of Ω directed symbols rather than symbols, since the elements of Σ are often referred to as symbols in the literature. However, in the sequel the only set of interest is Ω . Therefore, whenever we say symbol in the sequel, we mean directed symbol.

Definition 1 The *reflection* of symbol a , denoted by $\tilde{a}$, is the symbol that is equal to a in its first component and differs from a in its second component. The reflection of set A of symbols, denoted by $\tilde{A}$, is the set of reflected symbols of A . $\square$

Thus, by reflection an input symbol becomes an output symbol and vice versa.

Definition 2 For set A of symbols, iA denotes the set of input symbols of A and oA denotes the set of output symbols of A . $\square$

As said earlier, we characterize the way in which a process communicates with other processes by specifying the symbols it can send and receive. This determines how processes in a network “hook up” to one another. We call this set of symbols the *alphabet* of a process. We assume our links to be unidirectional, which means that a process can use a particular link either for sending or for receiving. Hence, an alphabet cannot just be any subset of Ω , but it is subject to the following restriction.

Definition 3 An *alphabet* A is any subset of Ω such that $A \cap \tilde{A} = \emptyset$. $\square$

Thus, the symbols of an alphabet differ in their first components. Next we formalize the way in which we specify the sequences of events that a process may engage in.

Definition 4 A *trace* is a finite-length sequence of symbols. The empty trace is denoted by ε . The reflection of trace t , denoted by $\tilde{t}$, is the trace obtained by reflecting all symbols of t . Trace s is called a *prefix* of trace t when $(\exists u :: su = t)$, where juxtaposition of traces denotes concatenation. The *length* of trace t , denoted by $\ell(t)$, is the number of symbols in t . The *number of occurrences* of symbol a in trace t is denoted by $\#_a t$. $\square$

A trace may be viewed as denoting a sequence of events, which have either occurred, when that trace describes a certain history, or can occur, when that trace specifies an allowed behavior of a process.

Definition 5 A *trace set* is a set of traces. A trace set is called *prefix-closed* when it contains all the prefixes of its traces. The reflection of trace set T , denoted by $\tilde{T}$, is the set of all reflections of traces of T . $\square$

A trace set specifies the possible sequences of events that a process can engage in. During the operation of a process specified by trace set T , there is a trace t of events engaged in thus far. We characterize a process by a directed trace structure.

Definition 6 A *directed trace structure*, or DTS, is a pair consisting of an alphabet and a prefix-closed trace set over that alphabet. For DTS S we denote its alphabet by $\mathbf{a}S$ and its trace set by $\mathbf{t}S$. As a pair, DTS S is sometimes written as $\langle \mathbf{a}S, \mathbf{t}S \rangle$. The input alphabet of S , denoted by $\mathbf{i}S$, is $\mathbf{i}(\mathbf{a}S)$ and the output alphabet, denoted by $\mathbf{o}S$, is $\mathbf{o}(\mathbf{a}S)$. The reflection of S , denoted by $\tilde{S}$, has alphabet $(\mathbf{a}S)^\sim$ and trace set $(\mathbf{t}S)^\sim$.

The DTS *CHAOS* is defined as $\langle \emptyset, \emptyset \rangle$ and *QUIET* as $\langle \emptyset, \{\varepsilon\} \rangle$. The space of all DTSs is denoted by $\mathcal{DTS}$ and the subspace of all DTSs with alphabet A is denoted by $\mathcal{DTS}(A)$. $\square$

Notice that *CHAOS* and *QUIET* are the only DTSs with empty alphabet and also the only DTSs that are invariant under reflection.

So far for single processes. As said earlier, we want to study networks of processes. A network is a set of processes in which each link is used by at most two processes. At most one process can send signals on a certain link and at most one process can receive signals from that link. Links that are used by exactly one process allow communications of the network with the “outside world”, which in this paper is just another network. The latter network can be viewed as an environment of the former. Throughout this paper we refer to a network as a system. We also introduce an operator on systems to model their parallel composition in an operational sense.

Definition 7 A *system* X is a set of DTSs with pairwise disjoint alphabets, that is,

$$(\forall S, T : S \in X \wedge T \in X \wedge S \neq T : \mathbf{a}S \cap \mathbf{a}T = \emptyset).$$

Let A be the set of symbols $(\bigcup S : S \in X : \mathbf{a}S)$, called the system's symbol set. The sets $\mathbf{i}A$, $\mathbf{o}A$, and $A \setminus \tilde{A}$ are alphabets. These are called the *input*, *output*, and *external* alphabets of X respectively and they are denoted by $\mathbf{i}X$, $\mathbf{o}X$, and $\mathbf{e}X$ respectively. System X is called *closed* when $A = \tilde{A}$. The *pattern* of X , denoted by $\mathbf{p}X$, is the set of alphabets of the elements of X .

Notice that the union of two systems with disjoint symbol sets is a system. We do not consider the set of *internal* symbols $A \cap \tilde{A}$ to be an attribute of a system. That is, we do not distinguish systems that are equal modulo systematic renaming of internal links. We assume that in all cases internal symbols can be systematically renamed to fresh ones⁰.

The *operational parallel composition* of systems X and Y with disjoint external alphabets is denoted by $X \uplus Y$. Let X' and Y' be systems obtained from X and Y by consistently renaming their internal symbols such that the symbol sets of X' and Y' are disjoint. This renaming is possible because $\mathbf{e}X \cap \mathbf{e}Y = \emptyset$. We now define $X \uplus Y = X' \cup Y'$. This is a proper definition, i.e. independent of the chosen renaming, because the names of internal symbols are immaterial. In fact, we will assume, whenever we write $X \uplus Y$, that the symbol sets of X and Y are disjoint.

The space of all systems is denoted by $\mathcal{SYS}$. □

The empty set is a system and for DTS S the singleton $\{S\}$ is also a system. The external alphabet of the composite of systems X and Y satisfies

$$\mathbf{e}(X \uplus Y) = (\mathbf{e}X \cup \mathbf{e}Y) \setminus (\mathbf{e}X \cup \mathbf{e}Y)^\sim$$

and the symbols in $\mathbf{e}X \cap \widetilde{\mathbf{e}Y}$ and $\widetilde{\mathbf{e}X} \cap \mathbf{e}Y$ have become internal in the composite. For distinct DTSs S and T we have that $\{S, T\}$ forms a system if and only if $\{S\} \uplus \{T\}$ is defined, i.e. if and only if $\mathbf{a}S \cap \mathbf{a}T = \emptyset$, and in that case we have

$$\{S\} \uplus \{T\} = \{S, T\}.$$

It may be felt as counterintuitive that for finite systems the number of elements in the parallel composite need not be equal to the numbers of elements in the systems composed. For example, if we take for X the singleton system $\{QUIET\}$, then $X \uplus X$ is defined and equals X . The only elements that may be “absorbed” in this way are *CHAOS* and *QUIET*. That this is harmless will become clear from their role in the operation of a system.

⁰This assumption requires that external alphabets of systems are of sufficiently low cardinality compared to that of Σ . For example, if external alphabets are required to be finite than a countably infinite set Σ suffices.

A closed system has no external links and all communications take place internally. The pattern of a system specifies how the components are connected to one another. Notice that a system can contain both the DTSSs *CHAOS* and *QUIET*. If that is the case, then they are identified in the pattern of that system, since they both have an empty alphabet. Again, this is harmless and will be clarified by the role of the pattern in the operation of a system.

1.1 Operation of systems

The previous section sufficiently describes the objects we want to study. In the next paragraphs we define how systems operate. As far as operation is concerned we consider closed systems only, since an open system always operates in some context, together with which it forms a closed system.

The key notion is that of the state of a system, which is given by the sequence of events that each component has communicated thus far. Subsequently, we define how a closed system moves from one state to the next, describing the operation of a system as a labeled transition system.

Definition 8 For system X a *system state* v , or state for short, is a mapping from $\mathbf{p}X$ to Ω^* such that $(\forall A : A \in \mathbf{p}X : v.A \in A^*)$. The set of all states of X is denoted by $\mathbf{v}X$. The length of state v , denoted by $\ell(v)$, is the sum of the lengths of the individual traces of v . For state v on system X and for some $S \in X$ we denote by $v(\mathbf{a}S : +a)$ the state that differs from v only on $\mathbf{a}S$, where it has the value $(v.\mathbf{a}S)a$. Similarly, for $S \in X$ and $T \in X$, $S \neq T$, we denote by $v(\mathbf{a}S, \mathbf{a}T : +a, +b)$ the state that differs from v only in $\mathbf{a}S$ and $\mathbf{a}T$. In $\mathbf{a}S$ it has the value $(v.\mathbf{a}S)a$ and in $\mathbf{a}T$ the value $(v.\mathbf{a}T)b$. With $\#_a v$ we denote $(\text{SUM } S : S \in X : \#_a v.\mathbf{a}S)$. Notice that at most one term in this sum is positive. A state v on system X is called *safe* when $(\forall S : S \in X : v.\mathbf{a}S \in \mathbf{t}S)$. $\square$

Notice that the length of a state may be infinite even if all individual traces have finite lengths, viz. if the system is infinite.

Notational convention: A mapping can be viewed as a set of pairs. The first element in the pair is an element of the domain of the mapping, the second element is the image under that mapping. Therefore, if X and Y are systems with disjoint symbol sets and if state $x \in \mathbf{v}X$ and state $y \in \mathbf{v}Y$, then $x \cup y$ is a state on $X \cup Y$. By Definition 7 the elements of the domain of a state are disjoint alphabets. By Definition 8 we have for any pair, say $(\mathbf{a}S, s)$, that $s \in (\mathbf{a}S)^*$, which implies that the pair is determined by s if $s \neq \varepsilon$. Using this property we denote a state by the set of second elements of its pairs. Because there is only

one empty trace this causes no problems. Also notice that the trace corresponding to *CHAOS* or *QUIET* can only be ε , since $\emptyset^* = \{\varepsilon\}$. Hence, it is harmless that these DTSs are identified in the pattern of a system.

The operation of a closed system is described as moving from state to state according to some rules, which we discuss later, starting in the initial state. This is the state in which none of the components has sent or received any signals.

Definition 9 The *initial* state of system X is the state v on X such that $(\forall A : A \in \mathbf{p}X : v.A = \varepsilon)$. A state of a system is called *reachable* if it can be reached from the initial state by a succession of moves according to the rules. $\square$

Notice that for any system containing a DTS with empty trace set, the initial state is not safe.

We distinguish two different sets of rules to move from one state to the next. The first set of rules is called the synchronous game and the other one is called the asynchronous game.

The synchronous game resembles CSP in the sense that sending and receiving are one event. It differs from CSP in the sense that there is an asymmetry between sending and receiving. The sender of a signal is the cause for this event to take place. In the asynchronous game sending a signal and receiving it are two distinct events. The only restriction is that receiving a signal is preceded by sending it. In the next two definitions we characterize a successor relation on states both for the synchronous and for the asynchronous game. This successor relation determines how a system may evolve starting in its initial state.

Definition 10 *Synchronous Game*

For v and w states on a closed system X we call w a successor of v when

$$\begin{aligned} (\exists S, T, a : S \in X \wedge T \in X \wedge \tilde{a} \in \mathbf{o}S \wedge a \in \mathbf{i}T \wedge (v.\mathbf{a}S)\tilde{a} \in \mathbf{t}S \\ : w = v(S, T : +\tilde{a}, +a)). \end{aligned}$$

$\square$

In words, w is a successor of v when w differs from v in two places, $\mathbf{a}S$ and $\mathbf{a}T$. In these places the traces of w are the ones of v extended with a symbol a in one place and its reflection in the other one. That symbol is such that S can send it, i.e. the trace of S generated in this game thus far, being $v.\mathbf{a}S$, can be extended with a according to $\mathbf{t}S$. Moreover, the pattern of the system is such that T is the component to receive that signal, i.e. $\tilde{a} \in \mathbf{i}T$. It is important to notice that these two events can take place

if the sending end can produce that symbol according to its specification. The receiving end need not be ready for the reception, which may cause the resulting trace not to belong to the corresponding trace set. Due to the prefix-closedness of the trace sets under consideration, if a trace associated with a component does not belong to the trace set of that component then no extension of that trace will belong to that trace set either. Therefore, a component that has received an input which it could not according to its trace set, will in this game never send another signal. We come back to this phenomenon later.

Definition 11 *Asynchronous Game*

For v and w states on a closed system X we say that w is a successor of v when it is either an output successor or an input successor of v . It is an *output successor* of v when

$$(\exists S, a : S \in X \wedge a \in \mathbf{o}S \wedge (v.\mathbf{a}S)a \in \mathbf{t}S : w = v(S : +a)).$$

It is an *input successor* of v when

$$(\exists S, a : S \in X \wedge a \in \mathbf{i}S \wedge \#_{\bar{a}}v > \#_av.\mathbf{a}S : w = v(S : +a)).$$

□

In the asynchronous game a symbol can be sent under the same conditions as in the synchronous game. Reception of that symbol, however, is now a distinct event. For an input event to take place there should have been sent more symbols on the corresponding link than have been received thus far.

1.2 The Correctness Concern

For two systems X and Y we want to define when X is at least as “good” as Y , denoted by $X \text{ sat } Y$. If X is viewed as implementation and Y as specification, one usually says “ X satisfies Y ”. We will, however, not explicitly distinguish systems that serve as implementation or as specification.

The measure of goodness depends on the correctness criteria that are taken into consideration. We capture a correctness concern by a predicate $P(X, U)$, where X is the system under consideration and system U can be viewed as (testing) environment of X . $P(X, U)$ holds if and only if X has “no problems” with U .

Definition 12 For correctness concern P and for systems X and Y of DTSs we say that X satisfies Y , denoted by $X \text{ sat } Y$, when

$$(\forall U : P(Y, U) : P(X, U)),$$

where U ranges over the space of systems.

□

Property 0 The relation **sat** is a pre-order, that is, it is reflexive and transitive. $\square$

Generally, $P(X, U)$ is the conjunction of a number of predicates. One of these predicates will always be that $X \uplus U$ defines a closed system, i.e. $\mathbf{e}X = \mathbf{e}\widetilde{U}$, so that modules and environments “hook up” to one another correctly. Throughout the rest of the paper we choose as our additional correctness concern that the closed system $X \uplus U$ has absence of computation interference, which is defined in the following way.

Definition 13 Closed system X is said to have *absence of computation interference* when all reachable states are safe. We denote the predicate that X is closed *and* has absence of computation interference by **nsi** X when playing the synchronous game and by **nai** X when playing the asynchronous game. $\square$

Thus, $P(X, U) \equiv \mathbf{nai} X \uplus U$ in the asynchronous game. Notice that whether there is computation interference does not depend on the names of the internal symbols of the closed system.

A system has computation interference, when a move can extend a trace in a reachable state with an input symbol while the resulting trace is not a member of the corresponding DTS’s trace set. We note that the situation of computation interference is undesirable. This is motivated by an interpretation of the model as an abstraction of the behavior of digital electrical circuits. The situation of computation interference can be interpreted as corresponding to a state of the circuitry that is not in agreement with the assumptions of digital operation. In a sense, it is an undefined digital state. The games, however, are always well defined. But it is irrelevant what successors the rules of the game assign to unsafe states. We have chosen for some particular behavior once there is computation interference, viz. the interfered process will no longer be able to send outputs. Our choice is fairly arbitrary; it has as advantage over alternatives that it requires no special treatment of unsafe states when defining the successor relation.

Property 1 A closed system X has absence of computation interference in the synchronous game when the initial state is safe and when for all safe reachable states v

$$(\forall S, T, a : S, T \in X \wedge \tilde{a} \in \mathbf{o}S \wedge a \in \mathbf{i}T : (v.S)\tilde{a} \in \mathbf{t}S \Rightarrow (v.T)a \in \mathbf{t}T).$$

It has absence of computation interference in the asynchronous game when the initial state is safe and when for all safe reachable states v

$$(\forall T, a : T \in X \wedge a \in \mathbf{i}T : \#_{\tilde{a}} v > \#_a v.T \Rightarrow (v.T)a \in \mathbf{t}T).$$

□

We can confine ourselves in this property to the *safe* reachable states since any non-initial unsafe reachable state can only be reached by a transition from a safe to an unsafe state. Notice that in a given system a state reachable under the synchronous rules is also reachable under the asynchronous rules in twice the number of moves, but not the other way round. Hence, if a closed system has absence of computation interference in the asynchronous game, then it has absence of computation interference in the synchronous game. Notice also that any system containing a DTS with empty trace set has computation interference, since the initial state is not safe.

Given a **sat**-relation we can define an equivalence relation on the space of all systems. We call two systems equivalent when no environment can distinguish the two. This equivalence is called testing equivalence in [3].

Definition 14 Two systems X and Y of DTSs are called equivalent, denoted by $X \text{ equ } Y$, when $X \text{ sat } Y \wedge Y \text{ sat } X$. □

Property 2 Relation **equ** is an equivalence relation and for systems X and Y we have

$$X \text{ equ } Y \equiv (\forall U :: P(X, U) \equiv P(Y, U)).$$

In fact, if P satisfies

$$P(X \uplus Y, Z) \equiv P(X, Y \uplus Z),$$

and this is the case for the correctness concerns we study, then **equ** is even a congruence relation on the space $\langle \mathcal{SYS}, \text{sat}, \uplus \rangle$. That is, for systems X , X' , Y , and Y' such that $X \text{ equ } X'$ and $Y \text{ equ } Y'$ we have

$$\begin{aligned} X \text{ sat } Y &\equiv X' \text{ sat } Y', \\ X \uplus Y &\text{ equ } X' \uplus Y'. \end{aligned}$$

Proof That **equ** is an equivalence relation is obvious. We show the two congruence conditions. Because of symmetry it is sufficient for the first condition to prove an implication. Assuming $X \text{ sat } Y$ we derive $X' \text{ sat } Y'$ by computing for arbitrary system Z :

$$\begin{aligned} &P(Y', Z) \\ = &\{ Y \text{ equ } Y' \} \\ &P(Y, Z) \\ \Rightarrow &\{ X \text{ sat } Y \} \\ &P(X, Z) \end{aligned}$$

$$= \{ X \text{ equ } X' \} \\ P(X', Z$$

Furthermore, we derive for arbitrary system Z :

$$\begin{aligned} & P(X \uplus Y, Z) \\ = & \{ \text{assumed property of } P \} \\ & P(X, Y \uplus Z) \\ = & \{ X \text{ equ } X' \} \\ & P(X', Y \uplus Z) \\ = & \{ \text{assumed property of } P \} \\ & P(X' \uplus Y, Z) \end{aligned}$$

Thus, we have $X \uplus Y \text{ equ } X' \uplus Y$. The replacement of Y by Y' follows from commutativity of $\uplus$. $\square$

The following relation on DTSs turns out to be important.

Definition 15 We define the relation $\sqsubseteq$ on $\mathcal{DTS}$ by

$$S \sqsubseteq T \equiv (\forall U : \text{nsi} \{T, U\} : \text{nsi} \{S, U\}),$$

for DTSs S and T (U also ranges over $\mathcal{DTS}$). $\square$

Notice the resemblance to the synchronous **sat** relation, which uses arbitrary systems as environments, instead of singletons. Actually, synchronous **sat** and $\sqsubseteq$ are the same relation, but we do not prove that in this paper. We quote the following results from [7].

Theorem 0 For DTSs S and T we have

$$\begin{aligned} S \sqsubseteq T \equiv & (\varepsilon \in \mathbf{t}T \Rightarrow \mathbf{a}S = \mathbf{a}T \wedge \varepsilon \in \mathbf{t}S) \wedge \\ & (\forall s, a : s \in \mathbf{t}S \cap \mathbf{t}T \\ & : (a \in \mathbf{o}S \wedge sa \in \mathbf{t}S \Rightarrow sa \in \mathbf{t}T) \\ & \wedge (a \in \mathbf{i}S \wedge sa \in \mathbf{t}T \Rightarrow sa \in \mathbf{t}S)). \end{aligned}$$

$\square$

The relation $\sqsubseteq$ is reflexive and transitive, and we have $S \sqsubseteq T$ for any DTSs S and T with $\mathbf{t}T = \emptyset$. Consequently, it is not a partial order on $\mathcal{DTS}$, since it lacks antisymmetry: we have $S \sqsubseteq T$ and $T \sqsubseteq S$ whenever the trace sets of S and T are empty, even if $\mathbf{a}S \neq \mathbf{a}T$. The abundance of DTSs with empty trace set is the only “problem”. If we leave out all but one, say *CHAOS*, of these, then $\sqsubseteq$ is a partial order. We also have

Theorem 1 The space $\langle \mathcal{DTS}(A), \sqsubseteq \rangle$ is a complete lattice, with $\langle A, \emptyset \rangle$ as top and $\langle A, (\mathbf{i}A)^* \rangle$ as bottom. $\square$

Furthermore, we have the following duality property.

Property 3 For DTSs S and T we have

$$S \sqsubseteq T \wedge \mathbf{t}T \neq \emptyset \equiv \tilde{T} \sqsubseteq \tilde{S} \wedge \mathbf{t}S \neq \emptyset.$$

Hence, for DTSs S and T with non-empty trace sets we have

$$S \sqsubseteq T \equiv \tilde{T} \sqsubseteq \tilde{S}.$$

$\square$

Theorem 2 Let $X \subseteq \mathcal{DTS}(A)$. If $\langle A, \emptyset \rangle \notin X$ then we characterize the trace set of the least upper bound of X , $\mathbf{t}(\sqcup X)$, as follows.

- $\varepsilon \in \mathbf{t}(\sqcup X)$
- for trace s and symbols $a \in \mathbf{i}A$ and $p \in \mathbf{o}A$

$$sa \in \mathbf{t}(\sqcup X) \equiv s \in \mathbf{t}(\sqcup X) \wedge (\forall S : S \in X \wedge s \in \mathbf{t}S : sa \in \mathbf{t}S)$$

$$sp \in \mathbf{t}(\sqcup X) \equiv s \in \mathbf{t}(\sqcup X) \wedge (\exists S : S \in X \wedge s \in \mathbf{t}S : sp \in \mathbf{t}S)$$

If $\langle A, \emptyset \rangle \in X$ then the least upper bound of X is the top $\langle A, \emptyset \rangle$. $\square$

Notice that the condition $\langle A, \emptyset \rangle \notin X$ is equivalent to $(\forall S : S \in X : \varepsilon \in \mathbf{t}S)$. Furthermore, the Theorem gives $\sqcup \emptyset = \langle \mathbf{i}A, (\mathbf{i}A)^* \rangle$.

Property 4 For $X \subseteq \mathcal{DTS}(A)$ we have $\mathbf{t}(\sqcup X) \subseteq (\bigcup S : S \in X : \mathbf{t}S)$. $\square$

2 Analysis of the Space of Systems

In this section we study the space $\langle \mathcal{SYS}, \mathbf{sat} \rangle$ under the equivalence relation based on the asynchronous game. The composition operator $\uplus$ will be treated in the next section. Throughout the rest of the paper **equ** stands for this asynchronous equivalence. Thus, we are interested in $\langle \mathcal{SYS}, \mathbf{sat} \rangle / \mathbf{equ}$. Our goal is to find a simpler isomorphic space. We show that $\langle \mathcal{DTS}, \sqsubseteq \rangle$ contains such an isomorphic subspace. We do so by exhibiting a homomorphism $\llbracket \cdot \rrbracket$ from $\langle \mathcal{SYS}, \mathbf{sat} \rangle$ into $\langle \mathcal{DTS}, \sqsubseteq \rangle$ satisfying the condition

$$X \mathbf{equ} Y \equiv \llbracket X \rrbracket = \llbracket Y \rrbracket,$$

for all systems X and Y . This condition expresses that $\llbracket \cdot \rrbracket$ exactly identifies equivalent systems. That $\llbracket \cdot \rrbracket$ is a homomorphism means, as you may know, that it satisfies $X \text{ sat } Y \equiv \llbracket X \rrbracket \sqsubseteq \llbracket Y \rrbracket$. We will see that this already implies the above condition on $\llbracket \cdot \rrbracket$, because $\sqsubseteq$ is a partial order on the image space under $\llbracket \cdot \rrbracket$. The desired result now follows from the Homomorphism Theorem [10, p. 79]. We will also explicitly characterize the image space of $\llbracket \cdot \rrbracket$.

The definition of the homomorphism is based on the following non-trivial property: Each system of DTSs is equivalent to a suitable system consisting of a single DTS. We will even show that each **equ**-equivalence class of systems contains a very special singleton system. The DTS member of that special equivalent singleton system serves as image under the homomorphism. We now give the construction of this singleton system and we define the space of DTSs that will turn out to be the subspace of images.

Definition 16 For system X , its set of *friends*, denoted by $FR(X)$, is defined by

$$FR(X) = \{U : U \in \mathcal{DTS} \wedge \text{nai} \{U\} \uplus X : U\}.$$

The mapping $\llbracket \cdot \rrbracket$ from $\mathcal{SYS}$ to $\mathcal{DTS}$ is defined by

$$\llbracket X \rrbracket = \begin{cases} \text{CHAOS} & \text{if } FR(X) = \emptyset \\ (\sqcup FR(X))^\sim & \text{if } FR(X) \neq \emptyset \end{cases}$$

for all systems X . The space $\mathcal{DI}$ of *delay-insensitive*¹ DTSs is defined by

$$\mathcal{DI} = \{S : S \in \mathcal{DTS} \wedge \text{nai} \{S, \tilde{S}\} : S\} \cup \{\text{CHAOS}\}.$$

□

Notice that all friends of system X have the same external alphabet, viz. $\widetilde{\text{e}X}$, and that they have non-empty trace sets. Hence, on account of Theorem 2 the trace set of $\llbracket X \rrbracket$ is non-empty if and only if $FR(X)$ is non-empty and, thus, $\llbracket X \rrbracket = \text{CHAOS}$ if and only if $FR(X) = \emptyset$. It is instructive to verify that $FR(\emptyset) = \{\text{QUIET}\}$ and, hence, $\llbracket \emptyset \rrbracket = \text{QUIET}$.

Notice that $\langle \mathcal{DI}, \sqsubseteq \rangle$ is a poset since it contains only one DTS with empty trace set, viz. CHAOS , which is the top. One readily verifies that $\mathcal{DI}$ is closed under reflection. Notice also, if we have proved $\llbracket \cdot \rrbracket$ to be a homomorphism, that **sat** and $\sqsubseteq$ “coincide” on $\mathcal{DI}$, i.e. for $S \in \mathcal{DI}$ and $T \in \mathcal{DI}$ we have $\{S\} \text{ sat } \{T\} \equiv S \sqsubseteq T$.

We make the following claims:

- (a) $X \text{ equ } \{\llbracket X \rrbracket\}$,

¹The name ‘delay-insensitive’ was taken from [6].

$$(b) \llbracket X \rrbracket \in \mathcal{DI},$$

$$(c) \llbracket \{S\} \rrbracket = S \text{ for all } S \in \mathcal{DI}.$$

The latter two claims express that $\mathcal{DI}$ is the image space of $\llbracket \cdot \rrbracket$. Claims (a) and (c) express that the singleton-system constructor $\{\cdot\}$ is the inverse of $\llbracket \cdot \rrbracket$ modulo equivalence. Assuming these claims and monotonicity of **nai** (see below) to hold for a moment, we can show that $\llbracket \cdot \rrbracket$ satisfies the requirements mentioned earlier, viz. that it is a homomorphism which exactly identifies equivalent systems. First, to see that it is a homomorphism we prove two implications separately:

$$\begin{aligned} & X \text{ sat } Y \\ \Rightarrow & \quad \{ \text{definition of sat} \} \\ & \{ Z : \text{nai } Y \uplus Z : Z \} \subseteq \{ Z : \text{nai } X \uplus Z : Z \} \\ \Rightarrow & \quad \{ \text{set theory and definition of } FR(X) \} \\ & FR(Y) \subseteq FR(X) \\ \Rightarrow & \quad \{ \text{def. of } \llbracket \cdot \rrbracket \text{ and } \sqsubseteq, \text{ using Property 3; case analysis on } FR(Y) = \emptyset \} \\ & \llbracket X \rrbracket \sqsubseteq \llbracket Y \rrbracket \end{aligned}$$

and

$$\begin{aligned} & \llbracket X \rrbracket \sqsubseteq \llbracket Y \rrbracket \\ \Rightarrow & \quad \{ \text{monotonicity of nai (Theorem 3 below)} \} \\ & (\forall Z : \text{nai } \{\llbracket Y \rrbracket\} \uplus Z : \text{nai } \{\llbracket X \rrbracket\} \uplus Z) \\ = & \quad \{ \text{claim (a) and definition of equ} \} \\ & (\forall Z : \text{nai } Y \uplus Z : \text{nai } X \uplus Z) \\ = & \quad \{ \text{definition of sat} \} \\ & X \text{ sat } Y \end{aligned}$$

Second, to see that $\llbracket \cdot \rrbracket$ exactly identifies equivalent systems, i.e.

$$X \text{ equ } Y \equiv \llbracket X \rrbracket = \llbracket Y \rrbracket,$$

it is sufficient to notice that $\llbracket \cdot \rrbracket$ is a homomorphism and that $\sqsubseteq$ is a partial order on $\mathcal{DI}$, especially that it is antisymmetric.

The next subsection is devoted to the proofs of claims (a) through (c).

2.0 Proving the claims

Our first goal is claim (a). The case of a friendless system is dealt with in

Lemma 0 For system X we have

$$X \text{ equ } \{CHAOS\} \equiv \llbracket X \rrbracket = CHAOS.$$

Hence, if $FR(X) = \emptyset$, then

$$X \text{ equ } \{\llbracket X \rrbracket\}.$$

Proof We derive

$$\begin{aligned}
& X \text{ equ } \{CHAOS\} \\
= & \quad \{ \text{Property 2} \} \\
& (\forall Y :: \mathbf{nai} X \uplus Y \equiv \mathbf{nai} \{CHAOS\} \uplus Y) \\
= & \quad \{ \text{empty trace set implies computation interference; predicate calculus} \} \\
& (\forall Y :: \neg \mathbf{nai} X \uplus Y) \\
= & \quad \{ \text{see Note below} \} \\
& (\forall U : U \in \mathcal{DTS} : \neg \mathbf{nai} X \uplus \{U\}) \\
= & \quad \{ \text{definition of } FR(X) \} \\
& FR(X) = \emptyset \\
= & \quad \{ \text{property of } \llbracket X \rrbracket \} \\
& \llbracket X \rrbracket = CHAOS
\end{aligned}$$

Note: Because every singleton system is a system, we have the implication downwards. The implication upwards follows from the fact that if X has computation interference with $\{(\widetilde{\mathbf{e}X}, (\widetilde{\mathbf{o}X})^*)\}$ then it has interference with any system. $\square$

The other case, in which $FR(X) \neq \emptyset$, is proved by separately establishing $X \text{ sat } \{\llbracket X \rrbracket\}$ (Thm. 5) and $\{\llbracket X \rrbracket\} \text{ sat } X$ (Thm. 6). For the former we need monotonicity (Thm. 3) and continuity (Thm. 4) of $\mathbf{nai}$ with respect to $\sqsubseteq$, and a form of transitivity of $\mathbf{nai}$ (Lemma 3).

The following two Theorems express monotonicity and a form of upward continuity of $\mathbf{nai}$ with respect to $\sqsubseteq$. More precisely, we consider for system X the boolean-valued function which maps DTS S into $\mathbf{nai} \{S\} \uplus X$. On the booleans we take the order $\Leftarrow$, for which universal quantification corresponds to taking the least upper bound. First, however, we need a lemma on reachability.

Lemma 1 For DTSs S and T , such that $S \sqsubseteq T$, and for system X , such that $\mathbf{nai} \{T\} \uplus X$, we have

$$v \text{ is reachable in } \{S\} \uplus X \Rightarrow v \text{ is reachable in } \{T\} \uplus X.$$

Proof We prove the lemma by induction on the length of v . First of all observe that systems $\{S\} \uplus X$ and $\{T\} \uplus X$ have the same pattern, since $\mathbf{a}S = \mathbf{a}T$ follows from $S \sqsubseteq T$. Therefore, states on $\{S\} \uplus X$ and on $\{T\} \uplus X$ have the same domain.

Base: Because initial states are reachable by definition, the lemma holds for the state of length 0.

Step: Next we show that any successor v on $\{S\} \uplus X$ of a reachable state w in $\{S\} \uplus X$ is reachable in $\{T\} \uplus X$, given that the lemma holds for w , i.e.

$$w \text{ is reachable in } \{T\} \uplus X. \quad (0)$$

Input successors of w are solely determined by w and the pattern on which w is defined. Since $\mathbf{p}(\{S\} \uplus X) = \mathbf{p}(\{T\} \uplus X)$, we infer that each input successor of w on $\{S\} \uplus X$ is a successor of w on $\{T\} \uplus X$ as well.

Left to prove is that an output successor of w in $\{S\} \uplus X$ is a successor of w in $\{T\} \uplus X$. Therefore, assume that v is output successor of w in $\{S\} \uplus X$. In the transition from w to v either S or a component in X has made a move. If the latter is the case then that move can also be made in $\{T\} \uplus X$ by the rules of the game, since w is reachable in $\{T\} \uplus X$ on account of (0). If S has made an output move then we have for some symbol a

$$v = w(\mathbf{a}S : +a) \text{ and } a \in \mathbf{o}S \text{ and } (w.\mathbf{a}S)a \in \mathbf{t}S. \quad (1)$$

We have to show that v is a successor of w in $\{T\} \uplus X$. On account of (0) and (1) it suffices to show that $(w.\mathbf{a}T)a \in \mathbf{t}T$, using that $\mathbf{a}S = \mathbf{a}T$. We derive

$$\begin{aligned} & \text{true} \\ &= \{ (0) \text{ and } (1), \text{ using that } \mathbf{a}S = \mathbf{a}T \} \\ & \quad w.\mathbf{a}T \in \mathbf{t}T \wedge (w.\mathbf{a}T)a \in \mathbf{t}S \wedge a \in \mathbf{o}S \\ &\Rightarrow \{ S \sqsubseteq T \text{ by assumption, Theorem 0 } \} \\ & \quad (w.\mathbf{a}T)a \in \mathbf{t}T \end{aligned}$$

□

Theorem 3 *Monotonicity of nai*

For system X and for DTSs S and T we have

$$S \sqsubseteq T \Rightarrow (\mathbf{nai} \{T\} \uplus X \Rightarrow \mathbf{nai} \{S\} \uplus X).$$

Proof We assume $S \sqsubseteq T$ and

$$\mathbf{nai} \{T\} \uplus X \quad (0)$$

and prove $\mathbf{nai} \{S\} \uplus X$. We have to show that every reachable state v is safe in $\{S\} \uplus X$. First we show that we can confine our considerations to $v.\mathbf{a}S$ by proving that $(\forall U : U \in X : v.\mathbf{a}U \in \mathbf{t}U)$ for any reachable state v in $\{S\} \uplus X$. We derive

$$\begin{aligned}
& v \text{ is reachable in } \{S\} \uplus X \\
\Rightarrow & \quad \{ \text{Lemma 1} \} \\
& v \text{ is reachable in } \{T\} \uplus X \\
\Rightarrow & \quad \{ (0) \} \\
& v \text{ is safe in } \{T\} \uplus X \\
\Rightarrow & \quad \{ \text{definition of safety} \} \\
& (\forall U : U \in X : v.\mathbf{a}U \in \mathbf{t}U)
\end{aligned}$$

Hence, the remaining proof obligation is to show that $v.\mathbf{a}S \in \mathbf{t}S$ for every reachable state v in $\{S\} \uplus X$. We prove this by induction on the length of $v.\mathbf{a}S$.

Base:

$$\begin{aligned}
& \text{true} \\
= & \quad \{ (0), \text{rules of the game} \} \\
& \varepsilon \in \mathbf{t}T \\
\Rightarrow & \quad \{ S \sqsubseteq T, \text{using Theorem 0} \} \\
& \varepsilon \in \mathbf{t}S
\end{aligned}$$

Step: Any reachable state in which the component on $\mathbf{a}S$ is not the empty trace has come about by making a transition from reachable state v to $v(\mathbf{a}S : +a)$ in $\{S\} \uplus X$ for some symbol $a \in \mathbf{a}S$. Therefore, we assume that

$$v \text{ and } v(\mathbf{a}S : +a) \text{ are reachable in } \{S\} \uplus X. \quad (1)$$

for some $a \in \mathbf{a}S$. We show that $(v.\mathbf{a}S)a \in \mathbf{t}S$ given that

$$v.\mathbf{a}S \in \mathbf{t}S. \quad (2)$$

If $a \in \mathbf{o}S$ then $(v.\mathbf{a}S)a \in \mathbf{t}S$ by the rules of the game. Therefore, assume that

$$a \in \mathbf{i}S. \quad (3)$$

We derive

$$\begin{aligned}
& \text{true} \\
= & \quad \{ (1) \}
\end{aligned}$$

$$\begin{aligned}
& v(\mathbf{a}S : +a) \text{ is reachable in } \{S\} \uplus X \\
\Rightarrow & \quad \{ \text{Lemma 1} \} \\
& v(\mathbf{a}S : +a) \text{ is reachable in } \{T\} \uplus X \\
\Rightarrow & \quad \{ \mathbf{nai} \{T\} \uplus X \text{ on account of (0), using } \mathbf{a}S = \mathbf{a}T \} \\
& (v.\mathbf{a}S)a \in \mathbf{t}T \\
= & \quad \{ (3) \text{ and } (2) \} \\
& (v.\mathbf{a}S)a \in \mathbf{t}T \wedge v.\mathbf{a}S \in \mathbf{t}S \wedge a \in \mathbf{i}S \\
\Rightarrow & \quad \{ S \sqsubseteq T \text{ by assumption, using Theorem 0} \} \\
& (v.\mathbf{a}S)a \in \mathbf{t}S
\end{aligned}$$

□

An important consequence of monotonicity is that the set of friends is downward closed under $\sqsubseteq$.

Theorem 4 *Upward Continuity of nai*

For system X and non-empty set Z of DTSs with the same alphabet, we have

$$\mathbf{nai} \{\sqcup Z\} \uplus X \equiv (\forall U : U \in Z : \mathbf{nai} \{U\} \uplus X).$$

Proof Notice that the least upper bound of Z exists. The implication from left to right follows immediately from the monotonicity of $\mathbf{nai}$. We now concentrate on the implication from right to left. Assume the right-hand side and define E as $\sqcup Z$, that is, as the least upper bound of Z . We show that $\mathbf{nai} \{E\} \uplus X$ holds.

Since the right-hand side holds we have $(\forall U : U \in Z : \varepsilon \in \mathbf{t}U)$ and, hence, $\varepsilon \in \mathbf{t}E$ on account of Theorem 2. Since $Z \neq \emptyset$ we have that all components of X contain ε . Therefore, the initial state in $\{E\} \uplus X$ is safe, and reachable by definition. Next we assume

$$v \text{ is safe and reachable in } \{E\} \uplus X \tag{0}$$

and prove that any successor of v in $\{E\} \uplus X$ is safe in $\{E\} \uplus X$. This then proves $\mathbf{nai} \{E\} \uplus X$.

Output successors of a safe and reachable state are safe by the rules of the game. Hence, we confine our attention to input successors of v in $\{E\} \uplus X$. On account of Property 4 we infer that $\mathbf{t}E \subseteq (\bigcup U : U \in Z : \mathbf{t}U)$. Hence, we may assume $S \in Z$ to be such that

$$\mathbf{nai} \{S\} \uplus X \wedge v \text{ is reachable in } \{S\} \uplus X. \tag{1}$$

Input successors of v are solely determined by v and the pattern on which the game is played. Therefore, using (0) and (1), we infer that any input successor of v in $\{E\} \uplus X$ is reachable and safe in $\{S\} \uplus X$. Hence, if an input successor of v is the result of a move of a component of X then that successor is safe in $\{E\} \uplus X$. If an input successor of v is the result of a move of E then

$$v(E : +a) \text{ is an input successor of } v \text{ in } \{E\} \uplus X \quad (2)$$

for some

$$a \in \mathbf{i}E. \quad (3)$$

It remains to be proven that $(v.\mathbf{a}E)a \in \mathbf{t}E$. We derive

$$\begin{aligned} & \text{true} \\ = & \{ (0), (2), \text{ and the rules of the game } \} \\ & (\forall U : U \in Z \wedge v.\mathbf{a}E \in \mathbf{t}U : v(E : +a) \text{ is reachable in } \{U\} \uplus X) \\ \Rightarrow & \{ (\forall U : U \in Z : \mathbf{nai} \{U\} \uplus X) \text{ assumed } \} \\ & (\forall U : U \in Z \wedge v.\mathbf{a}E \in \mathbf{t}U : (v.\mathbf{a}E)a \in \mathbf{t}U) \\ \Rightarrow & \{ v.\mathbf{a}E \in \mathbf{t}E \text{ on account of (0), } a \in \mathbf{i}E \text{ on account of (3), and} \\ & E = \sqcup Z, \text{ using Theorem 2 } \} \\ & (v.\mathbf{a}E)a \in \mathbf{t}E \end{aligned}$$

□

An important consequence of continuity is that the set of friends has a $\sqsubseteq$ -maximum if it is non-empty:

Corollary 0 For system X such that $FR(X) \neq \emptyset$ we have $\mathbf{nai} \{ \llbracket X \rrbracket^\sim \} \uplus X$, i.e.

$$\llbracket X \rrbracket^\sim \in FR(X).$$

□

Combining the consequences of monotonicity and continuity yields for system X with $FR(X) \neq \emptyset$:

$$FR(X) = \{ U : U \sqsubseteq \llbracket X \rrbracket^\sim : U \}.$$

Lemma 2 Let E be a DTS and let X and Y be systems such that $\mathbf{nai} \{E\} \uplus X$ and $\mathbf{nai} \{\tilde{E}\} \uplus Y$. Then we have for states x on X and y on Y such that $x \cup y$ is reachable in $X \uplus Y$:

$$(\exists s : s \in \mathbf{t}E : \{s\} \cup x \text{ is reachable in } \{E\} \uplus X \wedge \\ \{\tilde{s}\} \cup y \text{ is reachable in } \{\tilde{E}\} \uplus Y).$$

Proof By induction on the length of $x \cup y$. On account of $\mathbf{nai} \{E\} \uplus X$ we have $\varepsilon \in \mathbf{t}E$. Therefore, the lemma holds when $x \cup y$ is the initial state in $X \uplus Y$.

Now let x be a state on X and let y be a state on Y such that $x \cup y$ is reachable in $X \uplus Y$ and such that the lemma holds, i.e. let s be such that

$$\{s\} \cup x \text{ is reachable in } \{E\} \uplus X \quad (0)$$

and

$$\{\tilde{s}\} \cup y \text{ is reachable in } \{\tilde{E}\} \uplus Y. \quad (1)$$

We prove the lemma by showing the existence of a trace of E with the desired properties for any successor of $x \cup y$ in $X \uplus Y$. For reasons of symmetry, reflection being self-inverse, it suffices to consider only successors that differ from $x \cup y$ on X . Therefore, assume $S \in X$ and $a \in \mathbf{a}S$ to be such that

$$x(S : +a) \cup y \text{ is a successor of } x \cup y \text{ in } X \uplus Y. \quad (2)$$

If $a \in \mathbf{o}S$ or $a \in \mathbf{i}S \setminus \widetilde{\mathbf{o}E}$ then we conclude from (0), (2), and the rules of the game that $\{s\} \cup x(S : +a)$ is reachable in $\{E\} \uplus X$, which, together with (1), shows that s has the desired properties.

Now assume $a \in \mathbf{i}S \cap \widetilde{\mathbf{o}E}$. Then we have, due to $\mathbf{nai} \{E\} \uplus X$ and $\mathbf{nai} \{\tilde{E}\} \uplus Y$

$$a \in \mathbf{i}S \wedge \tilde{a} \in \mathbf{o}E \wedge a \in \mathbf{i}\tilde{E} \wedge \tilde{a} \in \mathbf{o}Y \quad (3)$$

and on account of (2) and the rules of the game

$$\#_a x < \#_{\tilde{a}} y. \quad (4)$$

If $\#_{\tilde{a}} s = \#_{\tilde{a}} y$ then $\{s\} \cup x(S : +a)$ is reachable in $\{E\} \uplus X$ on account of (0), (3), (4), and the rules of the game. Hence, also in this case s has the desired properties. Finally, assume, using that $\#_a \tilde{s} \leq \#_{\tilde{a}} y$ on account of (1) and (3),

$$\#_a \tilde{s} < \#_{\tilde{a}} y. \quad (5)$$

Then we derive that $s\tilde{a}$ suffices:

$$\begin{aligned} & \text{true} \\ &= \{ (1) \} \\ & \quad \{\tilde{s}\} \cup y \text{ is reachable in } \{\tilde{E}\} \uplus Y \\ &= \{ \text{rules of the game, using (5) and } a \in \mathbf{i}\tilde{E} \text{ on account of (3)} \} \end{aligned}$$

$$\begin{aligned}
& \{\tilde{s}a\} \cup y \text{ is reachable in } \{\tilde{E}\} \uplus Y \\
= & \{ \mathbf{nai} \{\tilde{E}\} \uplus Y \text{ by assumption, } \tilde{s}a \text{ is a trace over } \mathbf{a}\tilde{E} \} \\
& s\tilde{a} \in \mathbf{t}E \wedge \{\tilde{s}a\} \cup y \text{ is reachable in } \{\tilde{E}\} \uplus Y \\
= & \{ \text{rules of the game, using (0) and } \tilde{a} \in \mathbf{o}E \wedge a \in \mathbf{i}S \text{ on account} \\
& \text{of (3)} \} \\
& s\tilde{a} \in \mathbf{t}E \wedge \{s\tilde{a}\} \cup x(S : +a) \text{ is reachable in } \{E\} \uplus X \\
& \wedge \{\tilde{s}a\} \cup y \text{ is reachable in } \{\tilde{E}\} \uplus Y
\end{aligned}$$

□

Lemma 3 For DTS E and systems X and Y such that $\mathbf{nai} \{E\} \uplus X$ and $\mathbf{nai} \{\tilde{E}\} \uplus Y$ we have $\mathbf{nai} X \uplus Y$.

Proof It is obvious that $X \uplus Y$ is a closed system if $\mathbf{nai} \{E\} \uplus X$ and $\mathbf{nai} \{\tilde{E}\} \uplus Y$. Let x be a state on X and let y be a state on Y such that $x \cup y$ is reachable in $X \uplus Y$. On account of Lemma 2 we may assume $s \in \mathbf{t}E$ to be such that $\{s\} \cup x$ is reachable in $\{E\} \uplus X$ and such that $\{\tilde{s}\} \cup y$ is reachable in $\{\tilde{E}\} \uplus Y$. Since $\mathbf{nai} \{E\} \uplus X$ and $\mathbf{nai} \{\tilde{E}\} \uplus Y$ we conclude that x is safe in X and that y is safe in Y . Hence, $x \cup y$ is safe in $X \uplus Y$. □

Theorem 5 For system X such that $FR(X) \neq \emptyset$, we have

$$X \text{ sat } \{\llbracket X \rrbracket\}.$$

Proof For system Y we derive

$$\begin{aligned}
& \mathbf{nai} \{\llbracket X \rrbracket\} \uplus Y \\
= & \{ \text{Corollary 0, using } FR(X) \neq \emptyset \} \\
& \mathbf{nai} \{\llbracket X \rrbracket^\sim\} \uplus X \wedge \mathbf{nai} \{\llbracket X \rrbracket\} \uplus Y \\
\Rightarrow & \{ \text{Lemma 3} \} \\
& \mathbf{nai} X \uplus Y
\end{aligned}$$

□

Our next aim is to establish $\{\llbracket X \rrbracket\} \text{ sat } X$ if $FR(X) \neq \emptyset$. We need three lemmata. Lemma 4 considers the interference-free closed system $X \uplus \{\llbracket X \rrbracket^\sim\}$ (Corr. 0) and it says that all input extensions of traces in $\llbracket X \rrbracket^\sim$ are really needed to avoid computation interference. Lemma 5 can be viewed as a converse of Lemma 2. It expresses a form of transitivity of reachability. Finally, Lemma 6 gives a condition under which the trace set of a DTS in a system can be extended without introducing interference.

Lemma 4 For system X such that $FR(X) \neq \emptyset$ we have

$$(\forall s : s \in \mathbf{t}[X]^\sim : (\exists x : x \text{ on } X : \{s\} \cup x \text{ is reachable in } \{[X]^\sim\} \uplus X)).$$

Proof Let $E = [X]^\sim$, that is, $E = \sqcup FR(X)$. By induction on the length of the traces in E .

Base: The initial state is reachable in any game.

Step: Let $a \in \mathbf{a}E$ and let sa be a trace in $\mathbf{t}E$. Then s is a trace in $\mathbf{t}E$, since $\mathbf{t}E$ is prefix-closed on account of the definition of $\sqcup$. For s we may assume a state x on X to exist such that $\{s\} \cup x$ is reachable in $\{E\} \uplus X$ by the induction hypothesis. For $a \in \mathbf{o}E$ the state $\{sa\} \cup x$ is reachable in $\{E\} \uplus X$ by the rules of the game.

For $a \in \mathbf{i}E$ we infer from $sa \in \mathbf{t}E$ that $(\forall U : \mathbf{nai}\{U\} \uplus X \wedge s \in \mathbf{t}U : sa \in \mathbf{t}U)$ by the definitions of E and least upper bound. Taking for U the DTS $F = \langle \mathbf{a}E, \mathbf{t}E \setminus \{sa\}(\mathbf{a}E)^* \rangle$, for which $s \in \mathbf{t}F$ and $sa \notin \mathbf{t}F$, we find $\neg \mathbf{nai}\{F\} \uplus X$. We have $\mathbf{nai}\{E\} \uplus X$ on account of Corollary 0. Therefore, the interference in $\{F\} \uplus X$ can only be due to a move that changes s into sa . Hence, there is a state y on X such that $\{s\} \cup y$ is safe and reachable in $\{F\} \uplus X$ with (non-safe) successor $\{sa\} \cup y$. Hence, we have $\#_{\bar{a}}y > \#_as$. Since $s \in \mathbf{t}E$ we also have that $\{s\} \cup y$ is reachable in $\{E\} \uplus X$, and therefore that $\{sa\} \cup y$ is reachable in $\{E\} \uplus X$ as well. $\square$

Lemma 5 Let E be a DTS and let X and Y be systems. For t a trace in E and for x a state on X and for y a state on Y such that $y \cup \{\tilde{t}\}$ is reachable in $Y \uplus \{\tilde{E}\}$ and $\{t\} \cup x$ is reachable in $\{E\} \uplus X$ we have that $y \cup x$ is reachable in $Y \uplus X$.

Proof By induction on the sum of the lengths of y , t , and x .

Base: $\ell(y) = \ell(t) = \ell(x) = 0$. Trivial.

Step: Assume

$$y \cup \{\tilde{t}\} \text{ is reachable in } Y \uplus \{\tilde{E}\} \tag{0}$$

and that

$$\{t\} \cup x \text{ is reachable in } \{E\} \uplus X. \tag{1}$$

We have that $\ell(y) + \ell(t) + \ell(x) > 0$. Therefore, if no predecessors of $y \cup \{\tilde{t}\}$ and $\{t\} \cup x$ exist that differ from these states in the components on $\tilde{E}$ and E respectively then a predecessor of $y \cup \{\tilde{t}\}$ exists which differs from this state in y or a predecessor of $\{t\} \cup x$ exists which differs from this state in x . Since reflection is self-inverse it suffices, by symmetry, to consider only one of the last two cases. Hence, we distinguish the cases (i) $y \cup \{\tilde{t}\}$ has a

predecessor differing on $\tilde{E}$ and $\{t\} \cup x$ has a predecessor differing on E , and (ii) $\{t\} \cup x$ has a predecessor differing in x .

Case (i): Trace t' and symbol a are such that $t'a = t$ and $\{y\} \cup \tilde{t}'$ and $\{t'\} \cup x$ are reachable in $Y \uplus \{\tilde{E}\}$ and $\{E\} \uplus X$ respectively. From the induction hypothesis we now immediately infer that $y \cup x$ is reachable in $Y \uplus X$.

Case (ii): State x' , DTS $S \in X$, and symbol a are such that

$$x = x'(S : +a) \text{ and } \{t\} \cup x' \text{ is reachable in } \{E\} \uplus X. \quad (2)$$

From (0), (2), and the induction hypothesis we infer

$$y \cup x' \text{ is reachable in } Y \uplus X. \quad (3)$$

If a does not come from E , i.e. $\tilde{a} \notin \mathbf{o}E$, then we conclude from (1), (2), and (3) that $y \cup x$ is reachable in $Y \uplus X$ by the rules of the game. In case $\tilde{a} \in \mathbf{o}E$ we know that $a \in \mathbf{i}S$ and we derive

$$\begin{aligned} & \text{true} \\ = & \{ \{t\} \cup x'(S : +a) \text{ and } \{t\} \cup x' \text{ are reachable in } \{E\} \uplus X \text{ on account} \\ & \text{of (1) and (2), } \tilde{a} \in \mathbf{o}E \} \\ & \#_{\tilde{a}t} > \#_a x'.S \\ \Rightarrow & \{ (0), \tilde{a} \in \mathbf{o}E, \text{ thus } \#_{\tilde{a}y} \geq \#_a \tilde{t} \} \\ & \#_{\tilde{a}y} > \#_a x'.S \\ \Rightarrow & \{ a \in \mathbf{i}S, y \cup x' \text{ is reachable in } Y \uplus X \text{ on account of (3) } \\ & y \cup x'(S : +a) \text{ is reachable in } Y \uplus X \\ = & \{ (2) \} \\ & y \cup x \text{ is reachable in } Y \uplus X \end{aligned}$$

□

Lemma 6 Let E be a DTS and let X and Y be systems such that $\mathbf{nai}\{E\} \uplus X$ and $\mathbf{nai}X \uplus Y$. For trace t in $\mathbf{t}E$, symbol $a \in \mathbf{o}E$, and state y on Y such that $\{\tilde{t}\tilde{a}\} \cup y$ is reachable in $\{\tilde{E}\} \uplus Y$ we have $\mathbf{nai}\{F\} \uplus X$, where $F = \langle \mathbf{a}E, \mathbf{t}E \cup \{ta\}(\mathbf{i}E)^* \rangle$. Furthermore, if $F \sqsubseteq E$ then $ta \in \mathbf{t}E$.

Proof We prove by induction that every reachable state in $\{F\} \uplus X$ is safe.

Base: The initial state of $\{F\} \uplus X$ is safe, due to $\mathbf{nai}\{E\} \uplus X$ and $\mathbf{t}E \subseteq \mathbf{t}F$.

Step: We assume that

$$v \text{ is a reachable and safe state in } \{F\} \uplus X. \quad (0)$$

Let w be a successor of v in $\{F\} \uplus X$. If w is an output successor of v , then w is safe by the rules of the game. Therefore, we can confine ourselves to the case in which

$$w \text{ is an input successor of } v \text{ in } \{F\} \uplus X. \quad (1)$$

We distinguish the cases $v.aF \in \mathbf{t}E$ and $v.aF \notin \mathbf{t}E$. In the first case we derive

$$\begin{aligned} & v.aF \in \mathbf{t}E \\ \Rightarrow & \quad \{ (0) \text{ and the rules of the game } \} \\ & v \text{ is reachable in } \{E\} \uplus X \\ \Rightarrow & \quad \{ (1), \text{ using that an input successor of } v \text{ is solely determined by} \\ & \quad \text{the pattern and } v \} \\ & w \text{ is reachable in } \{E\} \uplus X \\ \Rightarrow & \quad \{ \text{premise: } \mathbf{nai} \{E\} \uplus X \} \\ & w \text{ is safe in } \{E\} \uplus X \\ \Rightarrow & \quad \{ \mathbf{t}E \subseteq \mathbf{t}F \} \\ & w \text{ is safe in } \{F\} \uplus X \end{aligned}$$

If $v.aF \notin \mathbf{t}E$ then we have on account of (0) and the definition of F that v can be written as $\{\tau\} \cup x$, where $\tau \in \mathbf{t}F$, so that $u \in (\mathbf{i}F)^*$, and x is a state of X . We rewrite assumption (0) about v :

$$\{\tau\} \cup x \text{ is safe and reachable in } \{F\} \uplus X \quad (2)$$

and we derive

$$\begin{aligned} & \text{true} \\ = & \quad \{ \text{assumption} \} \\ & \{\tilde{t}\tilde{a}\} \cup y \text{ is reachable in } \{\tilde{E}\} \uplus Y \\ \Rightarrow & \quad \{ u \in (\mathbf{i}F)^*, \text{ definition of } F \} \\ & \{\tilde{t}\tilde{a}\tilde{u}\} \cup y \text{ is reachable in } \{\tilde{F}\} \uplus Y \\ \Rightarrow & \quad \{ \text{Lemma 5, using (2)} \} \\ & x \cup y \text{ is reachable in } X \uplus Y \end{aligned} \quad (3)$$

We want to show that any input successor of $\{\tau\} \cup x$ in $\{F\} \uplus X$ is safe. An input successor of $\{\tau\} \cup x$ in $\{F\} \uplus X$ involving a symbol of $\mathbf{i}F$ extends τ with a symbol of $\mathbf{i}F$ resulting in a safe state in $\{F\} \uplus X$ by the definition of F . Therefore, we confine our attention to input successors of $\{\tau\} \cup x$ not involving a symbol of $\mathbf{i}F$, i.e. we assume

$$\{\tau\} \cup x(S : +b) \text{ is a successor of } \{\tau\} \cup x \text{ in } \{F\} \uplus X \quad (4)$$

for some $S \in X$ and $b \in \mathbf{i}S$. It suffices to show that $x(S : +b)$ is safe in X . If $\tilde{b} \notin \mathbf{o}F$ then this is obvious since in that case $x(S : +b) \cup y$ is a (safe) successor of $x \cup y$ in $X \uplus Y$ by the rules of the game, using (3). Hence, we assume

$$\tilde{b} \in \mathbf{o}F. \quad (5)$$

Now we derive

$$\begin{aligned} & \text{true} \\ = & \{ \text{premise of the lemma} \} \\ & \{\tilde{t}\tilde{a}\} \cup y \text{ is reachable in } \{\tilde{E}\} \uplus Y \\ \Rightarrow & \{ \tau u \in \mathbf{t}F \text{ and } u \in \mathbf{i}F^* \text{ on account of (2)} \} \\ & \{\tilde{t}\tilde{a}\tilde{u}\} \cup y \text{ is reachable in } \{\tilde{F}\} \uplus Y \\ \Rightarrow & \{ \tilde{b} \in \mathbf{o}Y, \text{ since } \{\tilde{F}\} \uplus Y \text{ is closed and } \tilde{b} \in \mathbf{o}F \text{ on account of (5)} \} \\ & \#_{\tilde{b}}y \geq \#_{\tilde{b}}\tau u \\ = & \{ \#_{\tilde{b}}\tau u > \#_{\tilde{b}}x.S \text{ on account of (4) and (5)} \} \\ & \#_{\tilde{b}}y > \#_{\tilde{b}}x.S \\ \Rightarrow & \{ \text{rules of the games, using (3) and } \tilde{b} \in \mathbf{i}S \text{ on account of (4) and (5)} \} \\ & x(S : +b) \cup y \text{ is reachable in } X \uplus Y \\ \Rightarrow & \{ \mathbf{nai} X \uplus Y \text{ by assumption} \} \\ & x(S : +b) \text{ is safe in } X \end{aligned}$$

This completes the proof of $\mathbf{nai} \{F\} \uplus X$. Finally, if $F \sqsubseteq E$ then, because $ta \in \mathbf{t}F$, $t \in \mathbf{t}E$, and $a \in \mathbf{o}E$, we have $ta \in \mathbf{t}E$. $\square$

Theorem 6 For system X such that $FR(X) \neq \emptyset$, we have

$$\{\llbracket X \rrbracket\} \text{ sat } X.$$

Proof Let $E = \llbracket X \rrbracket$. Assuming $\mathbf{nai} X \uplus Y$ we prove $\mathbf{nai} \{E\} \uplus Y$. Let trace s over $\mathbf{a}E$ and state y on Y be such that

$$\{s\} \cup y \text{ is reachable in } E \uplus Y. \quad (0)$$

We show that $\{s\} \cup y$ is safe in $\{E\} \uplus Y$. First we show that y is safe in Y if $s \in \mathbf{t}E$.

In case $s \in \mathbf{t}E$ we may assume state x on X to be such that

$$\{\tilde{s}\} \cup x \text{ is reachable in } \{\tilde{E}\} \uplus X \quad (1)$$

on account of Lemma 4. From (0), (1) and Lemma 5 we conclude that $x \cup y$ is reachable in $X \uplus Y$. Since we have $\mathbf{nai} X \uplus Y$ by assumption, we have that y is safe in Y .

The only remaining proof obligation is to show that s in (0) is an element of $\mathbf{t}E$. Since $\varepsilon \in \mathbf{t}E$ this is the case initially and output moves of E do not violate this. Hence, we have to show if, in addition to (0), $s \in \mathbf{t}E$ and $\{sa\} \cup y$ is a successor of $\{s\} \cup y$ in $\{E\} \uplus Y$ for some $a \in \mathbf{i}E$ then $sa \in \mathbf{t}E$. This follows immediately from Lemma 6 since we have $\mathbf{nai} \{\tilde{E}\} \uplus X$ on account of Corollary 0, and $a \in \mathbf{i}E \equiv \tilde{a} \in \mathbf{o}\tilde{E}$ and $\tilde{E}$ is an upper bound so that $sa \in \mathbf{t}E$. $\square$

This finally takes care of claim (a):

Theorem 7 For system X we have

$$X \text{ equ } \{\llbracket X \rrbracket\}.$$

Proof In case $FR(X) = \emptyset$ then the equivalence follows from Lemma 0. In case $FR(X) \neq \emptyset$ the equivalence follows from Theorems 5 and 6. $\square$

We now get the proof for claim (b) almost for free:

Theorem 8 For system X with $FR(X) \neq \emptyset$, we have $\mathbf{nai} \{\llbracket X \rrbracket, \llbracket X \rrbracket^\sim\}$. Therefore, for any system X we have $\llbracket X \rrbracket \in \mathcal{DI}$.

Proof On account of Corollary 0 we know $\mathbf{nai} \{\llbracket X \rrbracket^\sim\} \uplus X$ if $FR(X) \neq \emptyset$. Theorem 6 and the definition of $\mathbf{sat}$ yield the result desired. Hence, in this case $\llbracket X \rrbracket \in \mathcal{DI}$. Otherwise, if $FR(X) = \emptyset$, then by definition $\llbracket X \rrbracket = \text{CHAOS} \in \mathcal{DI}$. $\square$

In order to prove claim (c) we need one more lemma:

Lemma 7 For DTSs U and S we have

$$\mathbf{nai} \{U, S\} \Rightarrow U \subseteq \tilde{S} \wedge \mathbf{t}S \neq \emptyset.$$

Proof Assume $\mathbf{nai} \{U, S\}$. Hence, S has non-empty trace set. We prove $U \subseteq \tilde{S}$ using Theorem 0. Since $\mathbf{nai} \{U, S\}$, we have $\mathbf{a}U = \mathbf{a}\tilde{S}$ and $\varepsilon \in \mathbf{t}U$. Furthermore, we derive for $s \in \mathbf{t}U \cap \mathbf{t}\tilde{S}$

$$\begin{aligned} & a \in \mathbf{o}U \wedge sa \in \mathbf{t}U \\ = & \{ \mathbf{a}U = \mathbf{a}\tilde{S}, \text{ rules of the game using } \tilde{s} \in \mathbf{t}S \} \\ & a \in \mathbf{i}S \wedge \{sa, \tilde{s}\} \text{ is reachable in } \{U, S\} \\ \Rightarrow & \{ \mathbf{nai} \{U, S\} \text{ by assumption } \} \end{aligned}$$

$$\begin{aligned}
& \tilde{s}\tilde{a} \in \mathbf{t}S \\
= & \{ \text{definition of } \sim \} \\
& sa \in \mathbf{t}\tilde{S}
\end{aligned}$$

and

$$\begin{aligned}
& a \in \mathbf{i}U \wedge sa \in \mathbf{t}\tilde{S} \\
= & \{ \text{rules of the game, using } a \in \mathbf{i}U \equiv \tilde{a} \in \mathbf{o}S \text{ and } s \in \mathbf{t}U \} \\
& a \in \mathbf{i}U \wedge \{s, \tilde{s}\tilde{a}\} \text{ is reachable in } \{U, S\} \\
\Rightarrow & \{ \mathbf{nai} \{U, S\} \text{ by assumption } \} \\
& sa \in \mathbf{t}U
\end{aligned}$$

□

Theorem 9 For DTSSs S and T in $\mathcal{DI}$ we have

$$\{S\} \mathbf{equ} \{T\} \Rightarrow S = T.$$

Therefore, we also have

$$\llbracket \{S\} \rrbracket = S.$$

Proof We derive for DTSSs S and T such that $\{S\} \mathbf{equ} \{T\}$:

$$\begin{aligned}
& S \in \mathcal{DI} \\
= & \{ \text{definition of } \mathcal{DI} \} \\
& \mathbf{nai} \{S, \tilde{S}\} \vee S = \mathbf{CHAOS} \\
= & \{ \{S\} \mathbf{equ} \{T\} \text{ assumed } \} \\
& \mathbf{nai} \{T, \tilde{S}\} \vee S = \mathbf{CHAOS} \\
\Rightarrow & \{ \text{Lemma 7 and Theorem 0} \} \\
& T \sqsubseteq S
\end{aligned}$$

Similarly, if $T \in \mathcal{DI}$ we have $S \sqsubseteq T$. The first result now follows from antisymmetry of $\sqsubseteq$ on $\mathcal{DI}$. For the second result we derive for $S \in \mathcal{DI}$:

$$\begin{aligned}
& \text{true} \\
= & \{ \text{Theorem 7} \} \\
& \llbracket \{S\} \rrbracket \mathbf{equ} \{S\} \\
\Rightarrow & \{ \text{Theorem 8, } S \in \mathcal{DI}, \text{ and above result} \} \\
& \llbracket \{S\} \rrbracket = S
\end{aligned}$$

□

This takes care of claims (a) through (c).

2.1 More results

Now that we have these main results it becomes fairly straightforward to prove other properties, without digging into the operational semantics over and over again. We give three examples.

Property 5 For systems X and Y we have

$$X \text{ sat } Y \equiv (\forall U : \text{nai } \{U\} \uplus Y : \text{nai } \{U\} \uplus X),$$

where U ranges over $\mathcal{DI}$.

Proof The implication from left to right follows from the definition of **sat**. Assuming the right-hand side we derive for system Z :

$$\begin{aligned} & \text{nai } Y \uplus Z \\ = & \quad \{ \text{Theorem 7} \} \\ & \text{nai } Y \uplus \{\llbracket Z \rrbracket\} \\ \Rightarrow & \quad \{ \text{assumption, using that } \llbracket Z \rrbracket \in \mathcal{DI} \} \\ & \text{nai } X \uplus \{\llbracket Z \rrbracket\} \\ = & \quad \{ \text{Theorem 7} \} \\ & \text{nai } X \uplus Z \end{aligned}$$

□

Property 6 For DTSs S and U such that $S \in \mathcal{DI}$ or $U \in \mathcal{DI}$ we have

$$\text{nai } \{U, S\} \equiv U \subseteq \tilde{S} \wedge \text{t}S \neq \emptyset.$$

Proof The implication from left to right follows from Lemma 7. Notice that on account of Property 3, the right-hand side is equivalent to

$$S \subseteq \tilde{U} \wedge \text{t}U \neq \emptyset.$$

Since $\mathcal{DI}$ is closed under reflection we need only consider the case where $S \in \mathcal{DI}$: the case $U \in \mathcal{DI}$ follows from symmetry and above remark. We now find the other implication by applying Theorem 3 with U for S , $\tilde{S}$ for T , and $\{S\}$ for X , using $\text{nai } \{S, \tilde{S}\}$ on account of $S \in \mathcal{DI}$ and $\text{t}S \neq \emptyset$. □

Property 7 For systems X and Y with $\llbracket Y \rrbracket \neq \text{CHAOS}$ we have

$$\text{nai } X \uplus Y \equiv \llbracket X \rrbracket \subseteq \llbracket Y \rrbracket^\sim.$$

Proof We derive

$$\begin{aligned} & \text{nai } X \uplus Y \\ = & \quad \{ \text{Theorem 7} \} \\ & \text{nai } \{\llbracket X \rrbracket, \llbracket Y \rrbracket\} \\ = & \quad \{ \text{Property 6, using Theorem 8 and } \llbracket Y \rrbracket \neq \text{CHAOS} \} \\ & \llbracket X \rrbracket \subseteq \llbracket Y \rrbracket^\sim \end{aligned}$$

□

3 Composition and Decomposition

In this section we study $\langle \mathcal{SYS}, \uplus \rangle$ under the congruence **equ**. We define a denotational parallel composition operator $\parallel$ for DTSs and show that $\langle \mathcal{SYS}, \uplus \rangle / \mathbf{equ}$ is isomorphic to $\langle \mathcal{DI}, \parallel \rangle$. We do this with the mapping $\llbracket \cdot \rrbracket$ from the previous section, which will also turn out to be a homomorphism in this setting. We also prove a number of other properties of composition.

Theorem 7 shows that system $\{S, T\}$ is equivalent to the system consisting of singleton system $\{\llbracket \{S, T\} \rrbracket\}$. It is obvious now how to define parallel composition.

Definition 17 The parallel composition of DTSs S and T with disjoint alphabets, denoted by $S \parallel T$, is defined as $\llbracket \{S, T\} \rrbracket$. $\square$

Remark: It is instructive to read the definition as $\llbracket \{S\} \uplus \{T\} \rrbracket$, which is readily interpreted as translating S and T from the space $\mathcal{DI}$ into $\mathcal{SYS}$, applying operational parallel composition, and subsequently translating the result back from $\mathcal{SYS}$ into $\mathcal{DI}$.

Property 8 For DTSs S and T we have $\{S \parallel T\} \mathbf{equ} \{S, T\}$. $\square$

That $\llbracket \cdot \rrbracket$ is a homomorphism is now readily shown. Considering systems X and Y we derive

$$\begin{aligned}
 & \llbracket X \uplus Y \rrbracket \\
 = & \quad \{ \text{Theorem 7 and } \mathbf{equ} \text{ is a congruence w.r.t. } \uplus \text{ (Property 2)} \} \\
 & \llbracket \llbracket X \rrbracket \uplus \llbracket Y \rrbracket \rrbracket \\
 = & \quad \{ \text{property of } \uplus \text{ for 2-process systems} \} \\
 & \llbracket \llbracket X \rrbracket, \llbracket Y \rrbracket \rrbracket \\
 = & \quad \{ \text{definition of } \parallel \} \\
 & \llbracket X \rrbracket \parallel \llbracket Y \rrbracket
 \end{aligned}$$

Composition has all the desired properties as we show in the next couple of theorems. Obviously, composition is commutative. Furthermore, both *CHAOS* and *QUIET* are composable with any other DTS S , and we have

$$\begin{aligned}
 S \parallel \text{CHAOS} &= \text{CHAOS}, \\
 S \parallel \text{QUIET} &= S.
 \end{aligned}$$

That is, *CHAOS* acts as zero and *QUIET* as unit of $\parallel$.

Theorem 10 $\parallel$ is monotonic under **sat** and under $\sqsubseteq$.

Proof Since sat and $\sqsubseteq$ coincide on $\mathcal{DT}$ it suffices to prove this theorem only for sat . Let $\{S\} \text{sat} \{T\}$. We show that $\{S \parallel U\} \text{sat} \{T \parallel U\}$ for any DTS U . This boils down to showing that for all systems V : $\text{nai} \{T \parallel U\} \uplus V \Rightarrow \text{nai} \{S \parallel U\} \uplus V$, by the definition of sat . We derive

$$\begin{aligned}
& \text{nai} \{T \parallel U\} \uplus V \\
= & \quad \{ \text{Property 8} \} \\
& \text{nai} \{T, U\} \uplus V \\
\Rightarrow & \quad \{ \{S\} \text{sat} \{T\}, \text{definition of sat} \} \\
& \text{nai} \{S, U\} \uplus V \\
= & \quad \{ \text{Property 8} \} \\
& \text{nai} \{S \parallel U\} \uplus V
\end{aligned}$$

□

Theorem 11 Parallel composition is associative if the external alphabets are pairwise disjoint.

Proof

$$\begin{aligned}
& S \parallel (T \parallel U) \\
= & \quad \{ \text{definition of } \parallel \} \\
& \llbracket \{S, (T \parallel U)\} \rrbracket \\
= & \quad \{ \text{Property 8, using that the alphabets of } S, T, \text{ and } U \text{ are disjoint} \} \\
& \llbracket \{S, T, U\} \rrbracket
\end{aligned}$$

Symmetry does the rest. □

Theorem 12 We have that $\parallel$ is continuous with respect to $\sqcup$ on $\mathcal{DTS}(A)$, that is, for any DTS P , alphabet A , and non-empty subset Z of $\mathcal{DTS}(A) \setminus \{\langle A, \emptyset \rangle\}$ we have

$$P \parallel \sqcup Z = \sqcup \{S : S \in Z : P \parallel S\}.$$

Proof Let P be a DTS and let Z be a non-empty subset of $\mathcal{DTS}(A)$ not containing $\langle A, \emptyset \rangle$ for some alphabet A . Hence, $\sqcup Z$ exists. Define Z' as $\{S : S \in Z : P \parallel S\}$, then Z' is a non-empty set of DTSs with the same alphabet. Hence, $\sqcup Z'$ also exists. We need to show that $P \parallel \sqcup Z$ is the least upper bound of Z' . We use the following defining property of least upper bound: U is least upper bound of set Z' if and only if

$$(\forall T :: (\forall V : V \in Z' : V \sqsubseteq T) \equiv U \sqsubseteq T).$$

Note that we need not consider DTSs T with empty trace set since they are greater than anything. For DTS T with non-empty trace set we derive

$$\begin{aligned}
& (\forall V : V \in Z' : V \sqsubseteq T) \\
= & \quad \{ \text{definition of } Z' \} \\
& (\forall S : S \in Z : P \parallel S \sqsubseteq T) \\
= & \quad \{ \text{Theorem 13 below (2nd version) using } \mathbf{t}T \neq \emptyset \} \\
& (\forall S : S \in Z : S \sqsubseteq (P \parallel \tilde{T})^\sim \wedge P \parallel \tilde{T} \neq \text{CHAOS}) \\
= & \quad \{ \text{predicate calculus, using } Z \neq \emptyset \} \\
& (\forall S : S \in Z : S \sqsubseteq (P \parallel \tilde{T})^\sim) \wedge P \parallel \tilde{T} \neq \text{CHAOS} \\
= & \quad \{ \text{property of least upper bound applied to } Z \} \\
& \sqcup Z \sqsubseteq (P \parallel \tilde{T})^\sim \wedge P \parallel \tilde{T} \neq \text{CHAOS} \\
= & \quad \{ \text{Theorem 13 below (2nd version) using } \mathbf{t}T \neq \emptyset \} \\
& P \parallel \sqcup Z \sqsubseteq T
\end{aligned}$$

□

Finally, we prove a factorization theorem. In practice we often encounter the problem, given a specification S and part M of its implementation, to find the rest of the implementation. Hence, we are asked to find a specification such that any X , satisfying that specification, composed with M satisfies S . The following theorem states that the specification $(M \parallel \tilde{S})^\sim$ is the strongest specification with that property on the space $\mathcal{DI}$. For practical reasons we do not state the theorem for DTSs in $\mathcal{DI}$ only, but generalize it to arbitrary DTSs. In practice we will not always be given the $\mathcal{DI}$ -representative but an arbitrary member of its class. Although we could, the following theorem shows that it is not always necessary to compute the $\mathcal{DI}$ -representative first.

Theorem 13 *Factorization Theorems*

For DTSs S, T, U we have

$$U \parallel S \sqsubseteq \tilde{T} \wedge \mathbf{t}T \neq \emptyset \equiv U \parallel T \sqsubseteq \tilde{S} \wedge \mathbf{t}S \neq \emptyset.$$

Hence, for DTSs S, M , and X we have

$$X \parallel M \sqsubseteq S \wedge \mathbf{t}S \neq \emptyset \equiv X \sqsubseteq (M \parallel \tilde{S})^\sim \wedge M \parallel \tilde{S} \neq \text{CHAOS}.$$

Therefore, for DTSs S, M , and X such that $M \parallel \tilde{S} \neq \text{CHAOS}$ we have

$$X \parallel M \sqsubseteq S \equiv X \sqsubseteq (M \parallel \tilde{S})^\sim.$$

Proof Notice that both the left-hand and the right-hand side of the first equivalence imply that the alphabets of S, T , and U are disjoint. We derive

$$\begin{aligned}
& U \parallel S \sqsubseteq \tilde{T} \wedge \mathbf{t}T \neq \emptyset \\
= & \{ \text{Property 6, using } U \parallel S \in \mathcal{DI} \} \\
& \mathbf{nai} \{ U \parallel S, T \} \\
= & \{ \text{Property 8, using that the alphabets of } S, T, \text{ and } U \text{ are disjoint} \} \\
& \mathbf{nai} \{ U \parallel T, S \} \\
= & \{ \text{Property 6, using } U \parallel T \in \mathcal{DI} \} \\
& U \parallel T \sqsubseteq \tilde{S} \wedge \mathbf{t}S \neq \emptyset
\end{aligned}$$

For the second equivalence we derive

$$\begin{aligned}
& X \parallel M \sqsubseteq S \wedge \mathbf{t}S \neq \emptyset \\
= & \{ \text{first equivalence} \} \\
& M \parallel \tilde{S} \sqsubseteq \tilde{X} \wedge \mathbf{t}X \neq \emptyset \\
= & \{ \text{Property 3} \} \\
& X \sqsubseteq (M \parallel \tilde{S})^\sim \wedge \mathbf{t}(M \parallel \tilde{S}) \neq \emptyset \\
= & \{ M \parallel \tilde{S} \in \mathcal{DI} \} \\
& X \sqsubseteq (M \parallel \tilde{S})^\sim \wedge M \parallel \tilde{S} \neq \text{CHAOS}
\end{aligned}$$

Finally, the third equivalence is an immediate consequence of the second, since the assumption $M \parallel \tilde{S} \neq \text{CHAOS}$ implies $\mathbf{t}S \neq \emptyset$. $\square$

4 Concluding Remarks

In this section we discuss various aspects of the work reported in this paper. We relate it to some other work in this area and mention ongoing and future research.

This work starts out from an operational semantics of networks of processes. We strongly believe that this is the right point of departure to get a handle on a formal and total definition of a process. On the one hand, these semantics can be viewed as a first abstraction of the operation of electrical circuits. This means that it is reasonable to assume that these formal networks of processes can indeed be implemented. On the other hand, it is formal and yet rich enough to be mathematically tractable. Using mathematical techniques we can abstract from detail which is irrelevant for a particular correctness concern. This correctness concern fully determines the corresponding denotational model. The only freedom left is the choice of that correctness concern. It is much easier to choose such a correctness concern than it is to come up with the corresponding denotational model.

As opposed to [5], where a similar approach is taken for undirected and synchronous communications, we have not introduced a process syntax, which we believe to be a separate issue. In [1] a similar denotational

space of synchronously communicating processes is defined, but no operational model is introduced. We expect our approach to be particularly useful when incorporating certain liveness properties. Rather than having to come up with a denotational model—for which we have an overwhelming number of possibilities—we have to come up with a correctness concern at the operational level, where the number of reasonable choices is easily seen to be quite limited. As in [3] we have the notion of testing equivalence. Our approach differs in the sense that the distinction between process and experimenter vanishes.

The results of this paper give a more elegant way to define the space of delay-insensitive specifications, which had been done before in [6, 2, 8]. In that setting it becomes clear why the term delay-insensitive has been chosen for the space DI . The fact that a DTS has no computation interference with its reflection is hardly a justification for the term delay-insensitive. The approach in this paper, however, demonstrates much more convincingly that the resulting space is *precisely* the result of the concern of computation interference. A new result is the general definition of composition of specifications and the factorization theorem to which this definition of composition leads.

The operational space of objects has been chosen so as to allow other correctness concerns to be incorporated as well. This means that we can apply the same techniques of abstraction as in this paper to obtain the abstract space under that correctness concern. The additional concern of transmission interference has been investigated in [9]. The approach advocated here turned out to be nicely applicable with that additional concern. We are currently investigating liveness properties. It turns out that imposing absence of lock, playing the symmetric synchronous game, results in an abstract space which is isomorphic to the refusal model in [4]. We are investigating absence of lock in the setting of this paper, i.e. as a concern in addition to absence of computation interference. We shall report on this in a subsequent paper. Within the same framework we have also started to look into more general liveness properties, but it is too early to report on it.

Acknowledgments

We express our thanks to the members of the Directed Specifications Group, whose involvement in numerous discussions has been very helpful. In particular, we would like to mention Rudolf Mak, Huub Schols, and Martin Waardenburg. Thanks are also due to the Computer Science Department of Washington University in St. Louis, which enabled us to start on this research project.

References

- [0] W. Chen, J. T. Udding, and T. Verhoeff, "Networks of Communicating Processes and Their (De)-Composition", *Proceedings of the Conference on the Mathematics of Program Construction*, Lecture Notes in Computer Science, vol. 375, Springer-Verlag, 1989 (to appear).
- [1] D. L. Dill, *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*, PhD thesis, C.S. Dept., Carnegie Mellon Univ., Pittsburgh PA, 1988.
- [2] J. C. Ebergen, *Translating Programs into Delay-Insensitive Circuits*, PhD thesis, Eindhoven Univ. of Techn., The Netherlands, 1987.
- [3] M. Hennessy, *Algebraic Theory of Processes*, The MIT Press, Series in Found. of Comp., Cambridge MA, 1988.
- [4] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice-Hall Int., London, 1985.
- [5] E.-R. Olderog and C. A. R. Hoare, "Specification-Oriented Semantics for Communicating Processes", *Acta Informatica*, vol. 23, pp. 9–66, 1986.
- [6] J. T. Udding, *Classification and Composition of Delay-Insensitive Circuits*, PhD thesis, Eindhoven Univ. of Techn., The Netherlands, 1984.
- [7] J. T. Udding and T. Verhoeff, *The mathematics of Directed Specifications*, Technical Report WUCS-88-20, Washington University, St. Louis, June 1988.
- [8] T. Verhoeff, *Notes on Delay-Insensitivity*, Master's thesis, Eindhoven Univ. of Techn., The Netherlands, 1985.
- [9] M. Waardenburg, *Composition and Classification of Components*, Master's thesis, Eindhoven Univ. of Techn., The Netherlands, 1989.
- [10] H. Ehrig and B. Mahr, *Fundamentals of Algebraic Specification 1*, EATCS Monographs on Th. Comp. Sci, vol. 6, Springer, 1985.

In this series appeared :

No.	Author(s)	Title
85/01	R.H. Mak	The formal specification and derivation of CMOS-circuits
85/02	W.M.C.J. van Overveld	On arithmetic operations with M-out-of-N-codes
85/03	W.J.M. Lemmens	Use of a computer for evaluation of flow films
85/04	T. Verhoeff H.M.J.L. Schols	Delay insensitive directed trace structures satisfy the foam rubber wrapper postulate
86/01	R. Koymans	Specifying message passing and real-time systems
86/02	G.A. Bussing K.M. van Hee M. Voorhoeve	ELISA, A language for formal specifications of information systems
86/03	Rob Hoogerwoord	Some reflections on the implementation of trace structures
86/04	G.J. Houben J. Paredaens K.M. van Hee	The partition of an information system in several parallel systems
86/05	Jan L.G. Dietz Kees M. van Hee	A framework for the conceptual modeling of discrete dynamic systems
86/06	Tom Verhoeff	Nondeterminism and divergence created by concealment in CSP
86/07	R. Gerth L. Shira	On proving communication closedness of distributed layers
86/08	R. Koymans R.K. Shyamasundar W.P. de Roever R. Gerth S. Arun Kumar	Compositional semantics for real-time distributed computing (Inf.&Control 1987)
86/09	C. Huizing R. Gerth W.P. de Roever	Full abstraction of a real-time denotational semantics for an OCCAM-like language
86/10	J. Hooman	A compositional proof theory for real-time distributed message passing
86/11	W.P. de Roever	Questions to Robin Milner - A responder's commentary (IFIP86)
86/12	A. Boucher R. Gerth	A timed failures model for extended communicating processes

86/13	R. Gerth W.P. de Roever	Proving monitors revisited: a first step towards verifying object oriented systems (Fund. Informatica IX-4)
86/14	R. Koymans	Specifying passing systems requires extending temporal logic
87/01	R. Gerth	On the existence of sound and complete axiomatizations of the monitor concept
87/02	Simon J. Klaver Chris F.M. Verberne	Federatieve Databases
87/03	G.J. Houben J.Paredaens	A formal approach to distributed information systems
87/04	T.Verhoeff	Delay-insensitive codes - An overview
87/05	R.Kuiper	Enforcing non-determinism via linear time temporal logic specification.
87/06	R.Koymans	Temporele logica specificatie van message passing en real-time systemen (in Dutch).
87/07	R.Koymans	Specifying message passing and real-time systems with real-time temporal logic.
87/08	H.M.J.L. Schols	The maximum number of states after projection.
87/09	J. Kalisvaart L.R.A. Kessener W.J.M. Lemmens M.L.P. van Lierop F.J. Peters H.M.M. van de Wetering	Language extensions to study structures for raster graphics.
87/10	T.Verhoeff	Three families of maximally nondeterministic automata.
87/11	P.Lemmens	Eldorado ins and outs. Specifications of a data base management toolkit according to the functional model.
87/12	K.M. van Hee and A.Lapinski	OR and AI approaches to decision support systems.
87/13	J.C.S.P. van der Woude	Playing with patterns, searching for strings.
87/14	J. Hooman	A compositional proof system for an occam-like real-time language

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

88/10	J.C. Ebergen	A Formal Approach to Designing Delay Insensitive Circuits
88/11	G.J. Houben J.Paredaens	A graphical interface formalism: specifying nested relational databases
88/12	A.E. Eiben	Abstract theory of planning
88/13	A. Bijlsma	A unified approach to sequences, bags, and trees
88/14	H.M.M. ten Eikelder R.H. Mak	Language theory of a lambda-calculus with recursive types
88/15	R. Bos C. Hemerik	An introduction to the category theoretic solution of recursive domain equations
88/16	C.Hemerik J.P.Katoen	Bottom-up tree acceptors
88/17	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	Executable specifications for discrete event systems
88/18	K.M. van Hee P.M.P. Rambags	Discrete event systems: concepts and basic results.
88/19	D.K. Hammer K.M. van Hee	Fasering en documentatie in software engineering.
88/20	K.M. van Hee L. Somers M.Voorhoeve	EXSPECT, the functional part.
89/1	E.Zs.Lepoeter-Molnar	Reconstruction of a 3-D surface from its normal vectors.
89/2	R.H. Mak P.Struik	A systolic design for dynamic programming.
89/3	H.M.M. Ten Eikelder C. Hemerik	Some category theoretical properties related to a model for a polymorphic lambda-calculus.
89/4	J.Zwiers W.P. de Roever	Compositionality and modularity in process specification and design: A trace-state based approach.
89/5	Wei Chen T.Verhoeff J.T.Udding	Networks of Communicating Processes and their (De-)Composition.
89/6	T.Verhoeff	Characterizations of Delay-Insensitive Communication Protocols.
89/7	P.Struik	A systematic design of a parallel program for Dirichlet convolution.