ISRN SICS/R--91/04--SE

A Hierarchy of Compositional Models of I/O-Automata

by

Bengt Jonsson

SICS research report
R91:04
ISSN 0283-3638

A Hierarchy of Compositional Models of I/O-Automata *

Bengt Jonsson
Swedish Institute of Computer Science, Stockholm †
and Dept. of Computer Systems, Uppsala University

SICS Research Report 91:04
February 1991

Abstract

A semantic model of computer systems is compositional if it adequately represents the behavior of the modeled systems in a context of other systems. A compositional model is thus a good basis for specifying and reasoning about systems in a modular fashion. I/O-automata is a class of communicating systems which can represent several types of asynchronously communicating systems, such as message-passing distributed systems, systems with broadcast communication, and systems with shared variables. In contrast to many other classes of communicating systems (e.g. in CCS or CSP), semantic models based only on traces (sequences of communication events) can in a compositional way represent safety, liveness and many other properties of I/O-automata. In this paper, we investigate which semantic models of I/O-automata are compositional, and which are not. The investigation is confined to models based on traces. We study a number of trace-based semantical models of I/O-automata, which differ in their capability to represent safety, liveness, termination, and divergence properties. The defined models can be naturally ordered into a hierarchy, according to how much information they convey about the modeled systems. The main contribution of the paper is an investigation of whether there are other compositional models between adjacent compositional models in our hierarchy. Surprisingly enough, we can prove that for several pairs of adjacent models in the hierarchy, the gap between the two models contains no other compositional model. For instance, the nonexistence of a compositional model in the gap between a model that represents safety properties and a model that represents liveness properties means that liveness properties cannot be only partially represented, if compositionality is desired. We indicate how our results can be applied to derive results about full abstraction.

*An extended abstract of this research report has appeared under the same title in the *Proceedings of the Symposium on Mathematical Foundations of Computer Science*, Banska Bystrica, Czechoslovakia, Aug. 1990, published as vol. 452 of Lecture Notes in Computer Science, Springer Verlag, pp. 347-354.
This work was supported in part by the Swedish Board for Technical Development (STU) under contract No. 89-01220P as part of Esprit BRA project SPEC, No. 3096

†Address: SICS, Box 1263, S-164 28 Kista, SWEDEN, E-mail:bengt@sics.se

1 Introduction

Semantical models of communicating systems has been a topic of intensive study in the last years (e.g. [BHR84, Hoa85, Mil89, dNH84, OH86]). A purpose of that study is a better understanding of how to describe and reason about the behavior of communicating systems, e.g. in methods for specification and verification. A semantic model should abstract from the internal activity of a system, describing only its externally observable behavior. It should also be *compositional*, meaning that the denotation of a composed system can be obtained using only the denotations of its components. In other words, a compositional model must adequatly represent a system's interaction with its environment. A compositional semantic model is a good basis for modular specification and reasoning about communicating systems. Indeed, several compositional proof systems have been proposed (where the proof of a composed system can be split into proofs of its components), based on compositional semantic models (e.g. [BM85, Jon85, MC81, NDGO86, Zwi89]).

This paper is concerned with semantic models of a class of communicating systems, called I/O-automata by Lynch and Tuttle [LT87] and called I/O-systems by Stark [Sta84] and in our earlier work [Jon85, Jon87, Jon90]. I/O-automata can represent several types of systems with asynchronous communication such as message-passing distributed systems, systems with broadcast communication, and systems with shared variables. I/O-automata communicate through synchronous events, as in CCS and CSP [Mil89, Hoa85], but the occurrence of an event is controlled by at most one of the participating components. Thus asynchronous communication is represented as a special case of synchronous communication. For example, the event of sending a message into an unbounded channel is a synchronous event which involves the sender and the channel, but which is controlled only by the sender; the event of writing into a shared variable of another system involves both the writing system and the system owning the variable, but which is controlled only by the writer. For I/O-automata, semantic models based only on traces (sequences of communication events) can adequately represent safety, liveness and many other properties in a compositional way [Mis84, Jon85]. This is in contrast with the situation for CCS and CSP where the possibility of communication deadlocks makes the inclusion of refusals or ready-sets necessary [BHR84, NDGO86, dNH84, OH86].

This paper aims at investigating which semantic models of I/O-automata are compositional and which models are not. We confine the investigation to models based on traces. Different kinds of traces can, depending on their precise definition, represent safety properties, liveness properties, and properties of termination and divergence. For instance, safety properties can be modeled by prefix-closed sets of finite traces, whereas liveness properties can be modeled by (possibly infinite) traces of completed executions of a system. We can thus define a number of trace-based models of I/O-automata, which can or cannot represent safety, liveness, termination, or divergence. For a particular problem at hand, where some of these properties are important, one can then recommend a suitable model. Some of the models defined in this way are compositional, and some are not. The compositional models can be naturally ordered in a hierarchy, according to how much information they provide about the modeled systems. We say that a model 2 is more informative than a model 1 if the denotation of any system in model 1 can be obtained from its denotation in model 2.

The models that we define are of course not the only trace-based models of I/O-automata. One can for instance define other kinds of traces, or mix different kinds of traces in a model. One could imagine that such models could be useful for reasoning about systems in particular applications. Above, we have motivated the importance of compositionality for a model. We shall therefore investigate whether there are any compositional models that lie in the "gap"

between two adjacent models in our hierarchy. Interestingly enough, we shall find that in many such gaps there are no compositional models of I/O-automata. Assume for example that model 1 represents safety properties, and that model 2 represents safety and liveness properties. In our hierarchy, model 2 is more informative than model 1, and the models are adjacent models in our hierarchy. It turns out that there is no compositional model in the gap between these two models. The nonexistence of a compositional model between model 1 and model 2 means that liveness properties cannot be represented in strictly less detail than they are in model 2, without sacrificing compositionality. We use the term "minimal proper inclusion" for such empty gaps. We establish several such "minimal proper inclusions" between trace models of I/O-automata.

Our hierarchy, together with the minimal proper inclusions, gives a partial but nevertheless rather informative picture of which compositional trace-based models of I/O-automata exist. It can therefore guide the search for a suitable model for a particular problem at hand. One could for instance be interested in reasoning about deadlock properties (as in [Ora89]) or about certain types of progress properties (as e.g. in [MCS82]).

Our hierarchy can also be used to derive results about full abstraction for semantic models. Assume that model 1 represent properties of systems that are relevant for a particular problem at hand. Intuitively, a model 2 is fully abstract with respect to a model 1 if model 2 has added precisely enough information to model 1 for attaining compositionality. Thus, model 2 combines in an optimal way the abstraction from irrelevant detail provided by model 1 and the requirement of compositionality. Given a non-compositional model somewhere in our hierarchy, one can obtain a fully abstract model by going up in the hierarchy to the closest compositional model.

In the next section, we present I/O-automata and the operations composition and abstraction. In Section 3 we present the general framework for relating models, define our models of I/O-automata, and prove that they are compositional. Section 4 is the main section of the paper, where we relate the models in a hierarchy and investigate the gaps between adjacent models. In Section 5 we indicate how our hierarchy can be applied to obtain results about full abstraction. Section 6 contains conclusions, comparison with related work, and directions for further research. Proofs of some theorems are collected in the appendix.

2 I/O-automata

I/O-automata is a class of labeled transition systems, which satisfy certain restrictions. Labeled transition systems is well-established as a basic semantic description of computing systems (see e.g. Plotkin [Plo81], Manna and Pnueli [MP81], Milner [Mil89]).

We assume the existence of a set of *communication events*, which does not include the *silent event* τ. These are intended to represent observable interactions, e.g. the transmission of messages, between components of a system or between the system and its environment.

Definition 2.1 *An* I/O-automaton *is a tuple* $\langle I, O, S, S^0, T, \mathcal{F} \rangle$, *where*

- I *is a set of communication events, called* input events.
- O *is a set of communication events, called* output events, *which is disjoint from* I.
- S *is a set of* states.
- $S^0 \subseteq S$ *is a nonempty set of* initial states *of the system,*

$T \subseteq S \times (I \cup O \cup \{\tau\}) \times S$ *is a set of* labeled transitions. *A labeled transition* $\langle s_1, e, s_2 \rangle \in T$ *is denoted by* $s_1 \xrightarrow{e} s_2$, *where* $s_1, s_2 \in S$ *and* $e \in (I \cup O \cup \{\tau\})$,

$\mathcal{F} \subseteq \mathcal{P}(T)$ *is a finite collection of* fairness sets. *Each fairness set is subset of* T,

which in addition satisfies

1. *For each state* $s \in S$ *and input event* $e \in I$ *there is a state* $s' \in S$ *such that* $s \xrightarrow{e} s' \in T$,
2. *No fairness set* $F \in \mathcal{F}$ *contains any transition* $s \xrightarrow{e} s'$ *for which* $e \in I$. □

Intuitively, input events are communication events that are controlled by the environment, whereas output events are controlled by the system. The input events can e.g. represent transmissions of messages from the environment to the system, and output events can represent transmissions of messages by the system itself, both between components of the system and to its environment. The behavior of an I/O-automaton is determined by its states, and how the state may change in transitions. Simultaneously with a transition, a communication event may occur, or else the transition is labeled with the silent event τ. The fairness sets give conditions for when and how a sequence of transitions must be continued. Intuitively, each fairness set is a set of transitions which must not be neglected indefinitely in an execution of the system. For instance, a fairness set could represent the transitions controlled by a certain processor of a system, which will continue executing as long as it is not blocked.

The two requirements on I/O-automata formalize the assumption that an I/O-automaton does not control the occurrence of input events: input events can occur at any state (requirement 1), and the system cannot guarantee their eventual occurrence in an execution (requirement 2).

A transition $s \xrightarrow{e} s'$ is called an *input transition*, *output transition*, or *silent transition*, depending on whether e is an input event, an output event, or the silent event τ. A transition $s \xrightarrow{e} s'$ is *enabled* in the state s. A fairness set F is *enabled* in s if a transition in F is enabled in s.

Definition 2.2 *Let* A *be the I/O-automaton* $\langle I, O, S, S^0, T, \mathcal{F} \rangle$. *A* computation *of* A *is a finite or infinite sequence of transitions*

$$s^0 \xrightarrow{e^1} s^1 \xrightarrow{e^2} \cdots \xrightarrow{e^n} s^n \xrightarrow{e^{n+1}} \cdots$$

which starts in an initial state $s^0 \in S^0$ *and satisfies*

1. *if the sequence is infinite, then for each fairness set* $F \in \mathcal{F}$, *if* F *is enabled in all but finitely many states of the sequence, then the sequence contains infinitely many occurrences of transitions in* F, *and*
2. *if the sequence is finite, then no fairness set* $F \in \mathcal{F}$ *is enabled in the last state.*

A partial computation *of* A *is a finite sequence of transitions which starts in an initial state* $s^0 \in S^0$. □

Intuitively, a computation is a finite or infinite sequence of transitions, which starts in an initial state and satisfies the conditions imposed by the fairness sets. The fairness used here is often referred to as weak fairness or justice [LAJ81, Fra86]. We could also have allowed the inclusion of strong fairness in the definition, but that would not affect the results in this paper. Note that

a computation can be finite although some non-input transitions, which are not in any fairness set, are enabled in its last state. This situation can be avoided by requiring that each output and internal transition must be in some fairness set. Such a requirement is not related to the results of this paper, so we do not consider it.

Example 2.3 We define an I/O-automaton which represents a simple FIFO buffer, which receives messages from a channel *in* and transmits them onto channel *out* in the same order. We assume that the messages belong to a set M. The reception of a message $m \in M$ on *in* is a communication event denoted by $in(m)$. The transmission of a message $m \in M$ over *out* is denoted by $out(m)$. The buffer is represented by the I/O-automaton $\langle I, O, S, S^0, T, \mathcal{F} \rangle$, where

$I = \{in(m) \mid m \in M\}$.

$O = \{out(m) \mid m \in M\}$.

$S = M^*$,

$S^0 = \{\langle\rangle\}$, the empty sequence,

T contains, for each state $s \in S$ and message $m \in M$, the transitions (we use $\bullet$ to denote concatenation)

$$s \xrightarrow{in(m)} s \bullet m \qquad \text{and} \qquad m \bullet s \xrightarrow{out(m)} s \ ,$$

$\mathcal{F} = \{F\}$, where F the set of transitions in T labeled by an event of form $out(m)$ for some $m \in M$.

Intuitively, a state of the I/O-automaton represents the sequence of messages that have been received on *in*, but not yet transmitted on *out*. Initially, the state is the empty sequence. A transition of form $s \xrightarrow{in(m)} s \bullet m$ represents the reception of the message m on channel *in* and causes m to be appended to the state. A transition of form $s \xrightarrow{out(m)} s \bullet m$ represents the transmission of the message m on channel *out* and causes m to be removed from the state, provided that m is the first message of the state. The fairness set contains all transitions in which a message is transmitted over *out*. This implies that each message in the buffer will eventually be transmitted on *out*. To verify that $\langle I, O, S, S^0, T, \mathcal{F} \rangle$ is indeed an I/O-automaton, we check the two conditions in Definition 2.1

1. Since for each state s and $m \in M$ there is a a transition of form $s \xrightarrow{in(m)} s \bullet m$, we conclude that the first condition is satisfied.
2. The fairness set F does not contain any transition labeled by some $in(m)$ for m inM.

□

We next define the following operations on I/O-automata:

- The *composition* of the I/O-automata $A_1, \ldots, A_k$, denoted by $A_1 \| \cdots \| A_k$, yields an I/O-automaton in which $A_1, \ldots, A_k$ are components.
- The *abstraction* of a set E of output events of a system A, denoted by $A \setminus E$, yields an I/O-automaton for which the events in E are unobservable to the environment.

Definition 2.4 *The I/O-automata* $A_i = \langle I_i, O_i, S_i, S_i^0, T_i, \mathcal{F}_i \rangle$ *for* $i = 1, \ldots, k$ *are* compatible *if* $O_i \cap O_j = \emptyset$ *whenever* $i \neq j$. *If* $A_1, \ldots, A_k$ *are compatible, then their* composition, *denoted* $A_1 \| \cdots \| A_k$, *is the I/O-automaton* $A = \langle I, O, S, S^0, T, \mathcal{F} \rangle$, *where*

$O = \bigcup_{i=1}^{k} O_i$, *i.e., the union of the sets of output events of the components,*

$I = \bigcup_{i=1}^{k} I_i - O$, *i.e., the set of input events which are not output events,*

$S = S_1 \times \cdots \times S_k$, *i.e., the cartesian product of the sets of states of the components. A state* $s \in S$ *is a* k*-tuple, denoted as* $s = \langle s_1, \ldots, s_k \rangle$ *where* $s_i \in S_i$,

$S^0 = S_1^0 \times \cdots \times S_k^0$,

T *contains*

1. *all transitions of form* $\langle s_1, \ldots, s_k \rangle \xrightarrow{\tau} \langle s'_1, \ldots, s'_k \rangle$ *such that for some* i *the transition* $s_i \xrightarrow{\tau} s'_i$ *is in* T_i, *and* $s_j = s'_j$ *for* $j \neq i$, *and*
2. *all transitions of form* $\langle s_1, \ldots, s_k \rangle \xrightarrow{e} \langle s'_1, \ldots, s'_k \rangle$ *with* $e \neq \tau$ *where* $s_i \xrightarrow{e} s'_i \in T_i$ *if* $e \in (I_i \cup O_i)$ *and* $s_j = s'_j$ *if* $e \notin (I_i \cup O_i)$,

$\mathcal{F}$ *is obtained as follows: for each fairness set* $F_i \in \mathcal{F}_i$, *there is a fairness set* $F \in \mathcal{F}$ *consisting of all transitions* $\langle s_1, \ldots, s_k \rangle \xrightarrow{e} \langle s'_1, \ldots, s'_k \rangle \in T$ *such that* $s_i \xrightarrow{e} s'_i \in F_i$ *and* $e \in (I_i \cup O_i)$, *and all transitions* $\langle s_1, \ldots, s_k \rangle \xrightarrow{\tau} \langle s'_1, \ldots, s'_k \rangle \in T$ *such that* $s_i \xrightarrow{\tau} s'_i \in F_i$. □

Intuitively, the state of $A = A_1 \| \cdots \| A_k$ is a tuple, whose components are the states of the systems $A_1, \ldots, A_k$. The transitions of A describe how this state changes. A transition of A corresponds to either (1) a transition of a component A_i with a silent event, which does not affect other components and is performed in isolation, or (2) a transition with a communication event, which affects all components that can perform this communication event, corresponding to a synchronous communication involving the concerned components. Note that each communication event is an output event of at most one component, and is hence controlled by at most one component. This follows from the requirement that $A_1, \ldots, A_k$ be compatible. The fairness sets of A represent sets of transitions that must not be neglected indefinitely in a computation. Each fairness set of some A_i is therefore mapped in a natural way onto a fairness set of A.

Definition 2.5 *Let* $A = \langle I, O, S, S^0, T, \mathcal{F} \rangle$ *be an I/O-automaton, and let* $E \subseteq O$ *be a set of output events of* A. *The* abstraction *of* E *in* A, *denoted* $A \setminus E$, *is the tuple*

$$A \setminus E = \langle I, (O - E), S, S^0, T \setminus E, \mathcal{F} \setminus E \rangle \ ,$$

where $T \setminus E$ *and* $\mathcal{F} \setminus E$ *are obtained from* T *and* $\mathcal{F}$ *by changing all labels of transitions, which are in* E, *to* τ. □

Intuitively, the effect of the abstraction operation is that the occurrence of a set of communication events can no longer be observed by the environment. Formally, we achieve this by replacing abstracted events by the silent event τ. For the results of this paper, it is sufficient to define the abstraction operation with respect to output events. We can then abstract events that are used for communication between components, since these correspond to output events. Abstraction of input events can be emulated by a combination of the composition and abstraction operation: first the input events are transformed into output events by composition with another I/O-automaton which has as output events the events that are to be abstracted away, then the events can be abstracted away as in Definition 2.5.

Example 2.6 As an example, consider the FIFO buffer defined in Example 2.3. Let Buf_1 be the buffer in Example 2.3 with the channel name *in* replaced by *link* and let Buf_2 be the buffer in Example 2.3 with the channel name *out* replaced by *link*. The parallel composition $Buf_1 \| Buf_2$ is the I/O-automaton $\langle I, O, S, S^0, T, \mathcal{F} \rangle$, where

$I = \{in(m) \mid m \in M\}$.

$O = \{out(m) \mid m \in M\} \cup = \{link(m) \mid m \in M\}$.

$S = M^* \times M^*$,

$S^0 = \{(\langle\rangle, \langle\rangle)\}$, the pair with empty sequences,

T contains, for each state $(s_1, s_2) \in S$ and message $m \in M$, the transitions (we use $\bullet$ to denote concatenation)

$$(s_1, s_2) \xrightarrow{in(m)} (s_1, (s_2 \bullet m)) \quad , \quad (s_1, (s_2 \bullet m)) \xrightarrow{link(m)} ((m \bullet s_1), s_2) \quad , \quad ((m \bullet s_1), s_2) \xrightarrow{out(m)} (s_1, s_2) \ ,$$

$\mathcal{F} = \{F\}$, where F the set of transitions in T labeled by an event of form $out(m)$ or $out(m)$ for some $m \in M$.

Intuitively, a state of $Buf_1 \| Buf_2$ is a pair of sequences of messages, one for each component of the composition. A transition labeled $in(m)$ represent the reception of message m to the component Buf_2, a transition labeled $link(m)$ represent the transfer of message m from Buf_2 to Buf_1, and a transition labeled $out(m)$ represent the transmission of message m from Buf_1. □

The following proposition establishes that the operations on I/O-automata defined above are well-defined.

Proposition 2.7

- *If the I/O-automata $A_1, \ldots, A_k$ are compatible, then their composition $A_1 \| \ldots \| A_k$ is an I/O-automaton.*
- *If E is a set of output events of the I/O-automaton A, then the abstraction $A \setminus E$ is an I/O-automaton.*

□

Proof: A simple check. □

3 Compositional Models of I/O-automata

In this section, we present the traces models of I/O-automata that will be studied in the paper, and prove that they are all compositional.

Definition 3.1 *A* model $[\![\cdot]\!]$ *(of I/O-automata) is a mapping that maps I/O-automata to some set. A model* $[\![\cdot]\!]$ *is* compositional *with respect to a certain set of (partial) operations OP if for each (partial) operation* $op \in OP$ *there is a (partial) operation* $op_{\mathcal{M}}$ *such that* $[\![op(A_1, \cdots, A_k)]\!] = op_{\mathcal{M}}([\![A_1]\!], \cdots, [\![A_k]\!])$ *for any* $A_1, \ldots, A_k$ *such that* $op(A_1, \ldots, A_k)$ *is defined. In the following, OP will consist of the composition operation, and the abstraction operations for any set E of communication events.* □

Examples of rather uninteresting compositional models is the (uninformative) model that maps all I/O-automata to the same denotation, and the model which maps any I/O-automaton to itself.

We shall define the semantic models of I/O-automata that will be studied in this paper. These models are all based on traces. We therefore begin by defining several different kinds of traces.

Definition 3.2 *Let A be the I/O-automaton* $\langle I, O, S, S^0, T, \mathcal{F} \rangle$.

- *A* (completed) trace *of A is the (finite or infinite) sequence of communication events (i.e., non-*τ *events) in a computation of A.*
- *A* partial trace *of A is the (finite) sequence of communication events in a partial computation of A.*
- *A* terminated trace *of A is the sequence of communication events in a finite computation of A.*
- *A* divergent trace *of A is a finite sequence of communication events which occurs in an infinite computation of A.*

Define $I(A) = I$, $O(A) = O$, *and* $E(A) = I(A) \cup O(A)$. *Define* $P(A)$ *as the the set of partial traces,* $T(A)$ *as the set of traces,* $Q(A)$ *as the set of terminated traces, and* $D(A)$ *as the set of divergent traces of A.* □

We will often refer to completed traces simply as traces. Intuitively, partial traces represent observations that can be made after an arbitrary finite time during a computation, whereas traces can only be observed after a completed computation, i.e., after an infinite stretch of time has passed. In order to observe terminated or divergent traces, one must also be able to observe when no more τ-transition of the system will occur. This can be done, e.g. by assuming a lamp which indicates when the system will not perform any more τ-transitions before the next communication event.

Example 3.3 As an example, consider the unbounded FIFO buffer of Example 2.3. A partial trace of the buffer is a finite sequence t of communication events such that for any prefix t' of t, the sequence of messages output in t' is a prefix of the sequence of messages input in t'. A trace of the buffer is a (finite or infinite) sequence t of communication events such that all prefixes of t are partial traces of the buffer, and such that the sequence of messages output in t is the same as the sequence of messages input in t. A terminated trace of the buffer is a trace of the buffer which is finite. There are no divergent traces of the buffer. However, suppose that the buffer in addition contains an internal clock, which indefinitely performs internal events that do not affect the external behavior. In this case all finite traces are divergent, and there are no terminated traces. □

We now define a number of models of I/O-automata. In each of the models, the denotation of an I/O-automaton is a tuple which contains the sets $I(A)$ and $O(A)$ together with a certain choice of the different kinds of trace-sets. The sets $I(A)$ and $O(A)$ are always included, since they are necessary for determining which operations are applicable to an I/O-automaton. The model $[\![\cdot]\!]_E$ is defined by letting the denotation $[\![A]\!]_E$ of A be the tuple $\langle I(A), O(A)\rangle$. Other models are defined by using combinations of the subscripts P, T, Q, and D to indicate which kinds of traces are included in the denotation of an I/O-automaton, in addition to the input and output events. For instance, the model $[\![\cdot]\!]_P$ is defined by letting the denotation $[\![A]\!]_P$ of A be the tuple $\langle I(A), O(A), P(A)\rangle$. The model $[\![A]\!]_{TQ}$ is defined by letting the denotation $[\![A]\!]_{TQ}$ of A be the tuple $\langle I(A), O(A), T(A), Q(A)\rangle$, etc.

The model $[\![\cdot]\!]_P$ with partial traces is similar to models defined in e.g. [BM85, MC81, Zwi89], whereas model $[\![\cdot]\!]_T$ with traces is similar to models defined in e.g. [Jon85, Mis84, LT87]. The model $[\![\cdot]\!]_{PQ}$ is studied in [Ora89].

Let us introduce some notation. For a sequence q of communication events, and a set E of communication events let $q\lceil_E$ denote the restriction of q to the set E, let $q \setminus E$ denotes the subsequence communication events of q that are not in E, let E^* (E^ω) denote the set of finite (infinite) sequences of events in E, and let $E^\dagger = E^* \cup E^\omega$.

Theorem 3.4 *The models $[\![\cdot]\!]_E$, $[\![\cdot]\!]_P$, $[\![\cdot]\!]_{PQ}$, $[\![\cdot]\!]_T$, $[\![\cdot]\!]_{TQ}$, $[\![\cdot]\!]_{TD}$, $[\![\cdot]\!]_{TQD}$, and $[\![\cdot]\!]_Q$, are compositional.* □

Proof Sketch: We shall prove that $[\![A_1 || \ldots || A_k]\!]$ can be defined in terms of $[\![A_1]\!], \ldots, [\![A_k]\!]$ and that $[\![A \setminus E]\!]$ can be defined in terms of $[\![A]\!]$. Let $A_1, \ldots, A_k$ be compatible I/O-automata, and let A be their composition $A_1 || \ldots || A_k$. Then we have that $I(A) = \bigcup_{i=1}^{k} I(A_i) - O(A)$ and that $O(A) = \bigcup_{i=1}^{k} O(A_i)$. Furthermore, we have

$$P(A) = \{t \in (E(A))^* \ : \ (\forall i) \ \ t\lceil_{E(A_i)} \in P(A_i)\}$$

$$T(A) = \{t \in (E(A))^\dagger \ : \ (\forall i) \ \ t\lceil_{E(A_i)} \in T(A_i)\}$$

$$Q(A) = \{t \in (E(A))^* \ : \ (\forall i) \ \ t\lceil_{E(A_i)} \in Q(A_i)\}$$

$$D(A) = \{t \in (E(A))^* \ : \ (\exists i) \ [t\lceil_{E(A_i)} \in D(A_i) \ \ \wedge \ \ (\forall j \neq i) \ \ t\lceil_{E(A_j)} \in (T(A_j) \cup D(A_j))]\}.$$

For the abstraction operation, assume that A is an I/O-automaton, and E is a subset of $O(A)$. Then, if we let Ξ stand for either I, O, P, Q, T, we have that $\Xi(A \setminus E) = \Xi(A) \setminus E$. Furthermore, we have

$$D(A \setminus E) = D(A) \setminus E \ \cup \ \{t \in T(A) \setminus E \ : \ t \text{ is infinite and } t \setminus E \text{ is finite}\}$$

It follows that for the models of the theorem, $[\![A_1 || \ldots || A_k]\!]$ can be defined in terms of $[\![A_1]\!], \ldots, [\![A_k]\!]$ and that $[\![A \setminus E]\!]$ can be defined in terms of $[\![A]\!]$, since each kind of traces kan be composed separately except that divergent traces require information about (completed) traces. A proof that the composition operations are correct is given in the appendix. Similar proofs can also be found in e.g. [Sta84, Jon87, LT87, Jon85]. □

Theorem 3.5 *The models* $[\![\cdot]\!]_D$, $[\![\cdot]\!]_{PD}$, $[\![\cdot]\!]_{QD}$, *and* $[\![\cdot]\!]_{PQD}$ *are not compositional.* □

Proof Sketch: The reason why these models are not compositional is that in order to obtain the divergent traces of an I/O-automaton $A \setminus E$, one must in some cases know all traces of A. Consider the I/O-automata A_1 and A_2 where $[\![A_1]\!]_T = \langle \emptyset, \{e\}, \{e\}^\dagger \rangle$ and $[\![A_2]\!]_T = \langle \emptyset, \{e\}, \{e\}^* \rangle$, i.e., the difference between the I/O-automata is that A_1 can perform an infinite sequence of e's which A_2 cannot. Now, $\langle\rangle$ will be a divergent trace of $A_1 \setminus \{e\}$ but not of $A_2 \setminus \{e\}$. However, A_1 and A_2 have the same denotations in all the models $[\![\cdot]\!]_D$, $[\![\cdot]\!]_{PD}$, $[\![\cdot]\!]_{QD}$, and $[\![\cdot]\!]_{PQD}$. Hence these models are not compositional under the abstraction operation. □

4 Comparison Between Models of I/O-automata

In this section, we state the main results of the paper. We relate the compositional models that were defined in Section 3. We also investigate the gaps between adjacent models in the hierarchy, and check whether or not there are any compositional models in those gaps. This section contains several results, some of which require rather long proofs. We therefore first in Section 4.1 present the results without proofs. The proofs are thereafter contained in subsequent sections, each containing one or a number of related proofs.

4.1 Overview of Main Results

This section contains an overview of the results in Section 4. We first define the comparison of models according to how much information they convey about I/O-automata.

Definition 4.1 *A model* $[\![\cdot]\!]_1$ *(of I/O-automata)* contains less information *than another model* $[\![\cdot]\!]_2$, *denoted* $[\![\cdot]\!]_1 \sqsubseteq [\![\cdot]\!]_2$, *if* $[\![A]\!]_2 = [\![B]\!]_2$ *implies* $[\![A]\!]_1 = [\![B]\!]_1$ *for all I/O-automata* A, B. *We use* $[\![\cdot]\!]_1 \cong [\![\cdot]\!]_2$ *to denote that* $[\![\cdot]\!]_1 \sqsubseteq [\![\cdot]\!]_2 \sqsubseteq [\![\cdot]\!]_1$, *and use* $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!]_2$ *to denote that* $[\![\cdot]\!]_1 \sqsubseteq [\![\cdot]\!]_2 \not\sqsubseteq [\![\cdot]\!]_1$. □

Intuitively, $[\![\cdot]\!]_1$ contains less information than $[\![\cdot]\!]_2$ if the denotation $[\![A]\!]_2$ of an I/O-automaton A contains enough information to find the denotation $[\![A]\!]_1$. We observe that $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!]_2$ if and only if both $[\![\cdot]\!]_1 \sqsubseteq [\![\cdot]\!]_2$ and there are I/O-automata C, D such that $[\![C]\!]_2 \neq [\![D]\!]_2$ and $[\![C]\!]_1 = [\![D]\!]_1$. In the remainder of this section, we shal for given models $[\![\cdot]\!]_1$ and $[\![\cdot]\!]_2$ be concerned with finding the compositional models $[\![\cdot]\!]$ such that $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_2$. In many cases there are no such models, and we provide a separate definition for that case.

Definition 4.2 *If* $[\![\cdot]\!]_1$ *and* $[\![\cdot]\!]_2$ *are models (of I/O-automata) such that* $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!]_2$, *then the relation* $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!]_2$, *is* a minimal proper inclusion *if there is no compositional model* $[\![\cdot]\!]$ *such that* $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_2$. □

As an example, we shall later prove that the relation $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_T$ is a minimal proper inclusion. The nonexistence of a compositional model between the model $[\![\cdot]\!]_P$ which represents safety properties, and the model $[\![\cdot]\!]_T$ which represents both safety and liveness properties means that liveness properties cannot be represented in strictly less detail than they are in the model $[\![\cdot]\!]_T$, without sacrificing compositionality.

We now summarize the main results of this paper.

Overview of Results: *The compositional models of Section 3 can be ordered, according to how much information they contain, into the hierarchy of Figure 1. With relation to Figure 1,*

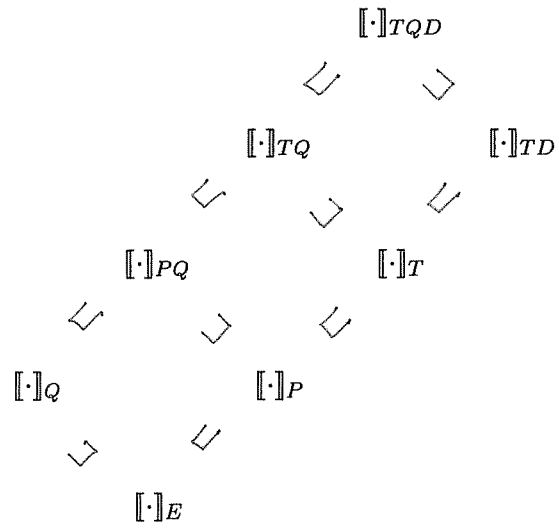

Figure 1: Relation between the models

the main results of the paper are the following

1. *The compositional models defined in Section 3 have relationships shown in Figure 1.*
2. *The relations* $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_Q$, $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_P$, $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_T$, $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_{PQ}$, *and* $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TQ}$ *are minimal proper inclusions.*
3. *There is exactly one model* $[\![\cdot]\!]$ *which satisfies* $[\![\cdot]\!]_Q \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_{PQ}$, *and exactly one model* $[\![\cdot]\!]$ *which satisfies* $[\![\cdot]\!]_{PQ} \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_{TQ}$.
4. *There are many models* $[\![\cdot]\!]$ *which satisfy* $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_{TD}$. □

We have not investigated the relations $[\![\cdot]\!]_{TD} \sqsubset [\![\cdot]\!]_{TQD}$ and $[\![\cdot]\!]_{TQ} \sqsubset [\![\cdot]\!]_{TQD}$.

Overview of Proofs: The first claim, that the relationships exist as in Figure 1, can be proven in a straightforward manner, using the obvious mappings between the models. In the remainder of Section 4, we will separate the proofs regarding the different inclusions into several subsections.

The proof that a certain proper inclusion is minimal requires in general a separate proof for each pair of models. Intuitively, the main part of a proof that $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!]_2$ is minimal consists in showing that if a model $[\![\cdot]\!]$ with $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!] \sqsubseteq [\![\cdot]\!]_2$ can distinguish between two given I/O-automata which are not distinguised by $[\![\cdot]\!]_1$, then one can build a context $\Phi[\cdot]$ which forces $[\![\cdot]\!]$ to distinguish between two arbitrary I/O-automata that are distinguished by $[\![\cdot]\!]_2$. On can then conclude that $[\![\cdot]\!] \cong [\![\cdot]\!]_2$. It is convenient to state a general lemma, which will be used in the conclusion of most proofs.

Lemma 4.3 *Let $[\![\cdot]\!]_1$ and $[\![\cdot]\!]_2$ be models of I/O-automata with $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!]_2$. Let h be a mapping from I/O-automata to I/O-automata such that $[\![h(C)]\!]_2 = [\![h(D)]\!]_2$ whenever $[\![C]\!]_1 = [\![D]\!]_1$. If for any I/O-automata A, B, C such that $[\![A]\!]_1 = [\![B]\!]_1$ and $[\![A]\!]_2 \neq [\![B]\!]_2$ there is a context $\Phi[\cdot]$ such that either*

$$[\![\Phi[A]]\!]_2 = [\![C]\!]_2 \qquad \text{and} \qquad [\![\Phi[B]]\!]_2 = [\![h(C)]\!]_2$$

or vice versa (i.e., $[\![\Phi[A]]\!]_2 = [\![h(C)]\!]_2$ and $[\![\Phi[B]]\!]_2 = [\![C]\!]_2$), then $[\![\cdot]\!]_1 \sqsubset [\![\cdot]\!]_2$ is a minimal proper inclusion. □

Intuitively, the function h maps all I/O-automata with the same denotation in $[\![\cdot]\!]_1$ to an I/O-automaton with the same constant denotation in the model $[\![\cdot]\!]_2$. If a model $[\![\cdot]\!]$ distinguishes between two I/O-automata with the same denotation in $[\![\cdot]\!]_1$, then there must be an I/O-automaton C such that $[\![C]\!] \neq [\![h(C)]\!]$. The conditions of the lemma imply that if $[\![C]\!] \neq [\![h(C)]\!]$ and $[\![\cdot]\!]$ is compositional then for arbitrary A and B with $[\![A]\!]_2 \neq [\![B]\!]_2$ we can conclude $[\![A]\!] \neq [\![B]\!]$.

Proof: Assume that there exists a model $[\![\cdot]\!]$ such that $[\![\cdot]\!]_1 \sqsubseteq [\![\cdot]\!] \sqsubset [\![\cdot]\!]_2$. We shall prove that in fact $[\![\cdot]\!]_1 \cong [\![\cdot]\!]$. By $[\![\cdot]\!] \sqsubset [\![\cdot]\!]_2$ there are I/O-automata A, B such that

$$[\![A]\!] = [\![B]\!] \qquad \text{and} \qquad [\![A]\!]_2 \neq [\![B]\!]_2$$

By the conditions of the lemma there is a context $\Phi[\cdot]$, such that $[\![\Phi[A]]\!]_2 = [\![C]\!]_2$ and $[\![\Phi[B]]\!]_2 = [\![h(C)]\!]_2$ (the vice versa case is handled analogously). If $[\![\cdot]\!]$ is compositional, we get by $[\![A]\!] = [\![B]\!]$ that $[\![\Phi[A]]\!] = [\![\Phi[B]]\!]$. This together with the conditions

$$[\![\Phi[A]]\!]_2 = [\![C]\!]_2 \qquad \text{and} \qquad [\![\Phi[B]]\!]_2 = [\![h(C)]\!]_2$$

and $[\![\cdot]\!] \sqsubset [\![\cdot]\!]_2$ gives that

$$[\![C]\!] = [\![\Phi[A]]\!] = [\![\Phi[B]]\!] = [\![h(C)]\!] \ .$$

Since this holds for arbitrary C and since $[\![h(C)]\!] = [\![h(D)]\!]$ whenever $[\![C]\!]_1 = [\![D]\!]_1$ we conclude that $[\![C]\!] = [\![D]\!]$ whenever $[\![C]\!]_1 = [\![D]\!]_1$, i.e., that $[\![\cdot]\!]_1 \cong [\![\cdot]\!]$. □

The remainder of Section 4 is organized as follows. In Section 4.2 we prove that the proper inclusion $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_P$ is minimal, in Section 4.3 we prove that $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_T$ is minimal, in Section 4.4 we prove that the proper inclusions $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_Q$, $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_Q$, and $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TQ}$ are minimal, in Section 4.5 we consider the inclusions $[\![\cdot]\!]_Q \sqsubset [\![\cdot]\!]_{PQ}$ and $[\![\cdot]\!]_{PQ} \sqsubset [\![\cdot]\!]_{TQ}$, and finally in Section 4.6 we present some compositional models between $[\![\cdot]\!]_T$ and $[\![\cdot]\!]_{TD}$.

4.2 Proof that $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_P$ is minimal

The central ingredient in the proof that $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_P$ is minimal is the construction in the following lemma. For disjoint sets I and O of communication events, let $ALL(I, O)$ be the I/O-automaton which has input events I, output events O, and can perform any sequence of communication events in $I \cup O$. Formally, $ALL(I, O)$ is the I/O-automaton $\langle I, O, \{s_{all}\}, \{s_{all}\}, T, \emptyset \rangle$ which has a single state s_{all}, no fairness sets, and where T contains the transition $s_{all} \xrightarrow{e} s_{all}$ for each $e \in (I \cup O)$.

Lemma 4.4 *Let A, B, and C be I/O-automata for which $[\![A]\!]_E = [\![B]\!]_E$. If there is a finite sequence of events $t \in E(A)^*$ such that $t \notin P(A)$ but $t \in P(B)$, then there is a context $\Phi[\cdot]$ such that $[\![\Phi[A]]\!]_P = [\![C]\!]_P$ and $[\![\Phi[B]]\!]_P = [\![ALL(I(C), O(C))]\!]_P$.* □

Proof: To begin with, we shall assume that the sets $E(A)$ and $E(C)$ are disjoint. In the proof, we construct an I/O-automaton N, such that the conclusion of the lemma is satisfied by the context $\Phi[\cdot] = (\cdot \| N) \setminus E(A)$. The idea of the construction is the following. The I/O-automaton N communicates both via the set of events $E(A)$ of A, and the events $E(C)$ of C. The I/O-automaton N communicates with the component that fills the hole of the context (in this case A or B), and records the sequence of communication events exchanged with A or B. Initially, N imitates the behavior of C with regard to the events in $E(C)$. However, if the sequence of events exchanged with the component that fills the hole in the context becomes t, then N can start to behave like $ALL(I(C), O(C))$.

A more detailed description of N can be given as the I/O-automaton $\langle I, O, S, S^0, T, \mathcal{F} \rangle$, where

$I = I(C) \cup O(A)$,

$O = O(C) \cup I(A)$,

S is the set of pairs $\langle t_A, s_C \rangle$ such that t_A is a finite sequence of events in $E(A)$ and s_C is a state of C, and the distinguished state s_{all}.

S^0 is the set of states $\langle \langle \rangle, s_C^0 \rangle$ such that s_C^0 is an initial state of C,

T is the set of transitions that are of one of the following forms:

(1) $\langle t_A, s_C \rangle \xrightarrow{e_A} \langle t_A \circ e_A, s_C \rangle$ where $e_A \in E(A)$ is a communication event exchanged with the I/O-automaton in the hole,

(2) $\langle t_A, s_C \rangle \xrightarrow{e_C} \langle t_A, s'_C \rangle$ where $s_C \xrightarrow{e_C} s'_C$ is a transition of C,

(3) $\langle t, s_C \rangle \xrightarrow{\tau} s_{all}$, where t is the partial trace that distinguishes A and B,

(4) $s_{all} \xrightarrow{e_C} s_{all}$ for any $e_C \in E(C)$,

$\mathcal{F} = \emptyset$.

To see that $\Phi[\cdot]$ really satisfies the conditions of the lemma, note that N can do any sequence of events that C can. Furthermore, N can imitate $ALL(I(C), O(C))$ in the case that the I/O-automaton that fills the hole in the context performs the partial trace t. Thus, $\Phi[\cdot]$ can emulate any partial computation of C regardless of which I/O-automaton fills the hole. Also, $\Phi[B]$ can perform any partial trace of $ALL(I(C), O(C))$ in a partial computation in which B has performed t. The conclusion of the lemma follows.

In the case where the sets $E(A)$ and $E(C)$ are not disjoint, we find two new sets of input and output events that are disjoint from $E(A)$ and $E(C)$, and repeat the construction of the lemma twice. □

Theorem 4.5 *The relation $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_P$ is a minimal proper inclusion.* □

Proof: The theorem follows from Lemma 4.4 and Lemma 4.3, using $h(C) = ALL(I(C), O(C))$, and the observation that whenever $[\![A]\!]_E = [\![B]\!]_E$ and $[\![A]\!]_P \neq [\![B]\!]_P$ there is a finite sequence of events $t \in (E(A))^*$ such that $t \notin P(A)$ but $t \in P(B)$ or vice versa. □

4.3 Proof that $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_T$ is minimal

The central ingredient in the proof that $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_T$ is minimal is the construction in the following lemma.

For a set P of finite sequences, let $\lceil P \rceil$ be the set containing P and in addition all infinite sequences t for which all finite prefixes of t are in P. For an I/O–automaton A, let $CHAOS(A)$ be an I/O-automaton for which $T(CHAOS(A)) = \lceil P(A) \rceil$. Intuitively, $CHAOS(A)$ is an I/O-automaton which has the same partial traces as A but can terminate after any partial trace, and also extend any chain of partial traces to an infinite trace. Formally, $CHAOS(A)$ can be represented by the I/O-automaton $\langle I(A), O(A), P(A), \{\langle\rangle\}, T, \emptyset\rangle$ where T contains the transition $t \xrightarrow{e} t \bullet e$ whenever $t \bullet e \in P(A)$.

Lemma 4.6 *Let A, B, and C be I/O-automata such that $[\![A]\!]_P = [\![B]\!]_P$. Assume that there is a finite or infinite sequence of events $t \in E(A)^\dagger$ such that $t \notin T(A)$ but $t \in T(B)$. Then there is a context $\Phi[\cdot]$ such that $[\![\Phi[A]]\!]_T = [\![C]\!]_T$ and $[\![\Phi[B]]\!]_T = [\![CHAOS(C)]\!]_T$.* □

Proof: To begin with, we shall assume that the sets $E(A)$ and $E(C)$ are disjoint. In the proof, we construct an I/O-automaton N, such that the conclusion of the lemma is satisfied by the context $\Phi[\cdot] = (\cdot \| N) \setminus E(A)$.

The idea of the construction is the following. The system N communicates both via the set of events $E(A)$ of A, and the events $E(C)$ of C. The system N records the sequence that has occurred on $E(A)$, and can behave both like C and like $CHAOS(C)$ on $E(C)$. The sequence that has occurred on $E(A)$ decides whether N should behave like C or like $CHAOS(C)$ on $E(C)$. The intention is that N shall behave like $CHAOS(C)$ if and only if the sequence of events on $E(A)$ in the completed computation is t.

In the beginning, N behaves like $CHAOS(C)$, and continues as long as the sequence of events on $E(A)$ is a prefix of t. Thereafter N switches to C if is seems likely that the sequence of events during the completed computation will not become t. This can occur if (1) the sequence of events on $E(A)$ is no longer a prefix of t, or (2) the sequence of events on $E(A)$ remains the same proper prefix of t. To cover case (1), N switches to C if an event occurs which makes the sequence on $E(A)$ no longer a prefix of t. To cover case (2), we introduce fairness sets which imply that N switches to C if the sequence on $E(A)$ stays a proper prefix of t indefinitely.

More precisely, the I/O-automaton N is defined as the tuple $\langle I, O, S, S^0, T, \mathcal{F}\rangle$, where

$I = O(A) \cup I(C)$

$O = I(A) \cup O(C)$

S contains (1) all states of C and (2) all tuples of the form $\langle t_A, t_C\rangle$ where t_A is a sequence in $(E(A))^*$ and $t_C \in P(C)$,

$S^0 = \{\langle\langle\rangle, \langle\rangle\rangle\}$,

T is the set of transitions that are of one of the following forms:

(1) $\langle t_A, t_C\rangle \xrightarrow{e_A} \langle t_A \bullet e_A, t_C\rangle$ where $e_A \in E(A)$ is a communication event exchanged with the I/O-automaton in the hole of the context,

(2) $\langle t_A, t_C\rangle \xrightarrow{e_C} \langle t_A, t_C \bullet e_C\rangle$ where $t_C \bullet e_C \in P(C)$,

(3) $\langle t_A, t_C \rangle \xrightarrow{\tau} s_C$ where s_C is a state of C which can occur at the end of some partial computation with t_C as sequence of communication events,

(4) $s_C \xrightarrow{e} s'_C$ whenever $s_C \xrightarrow{e} s'_C$ is a transition of C,

$\mathcal{F} = \{F_{even}, F_{odd}, F_{other}\} \cup \mathcal{F}_C$, where F_{even} is the set of transitions of form (3) in which t_A is a proper prefix of t with even length, F_{odd} is the set of transitions of form (3) in which t_A is a proper prefix of t with odd length, F_{other} is the set of transitions of form (3) in which t_A is not a prefix of t, and $\mathcal{F}_C$ are the fairness sets of C, applied to the transitions of form (4) that are transitions of C.

We must verify that N is indeed an I/O-automaton. Requirement (1) in definition 2.1 follows from the fact that an input event can always be added to $(E(A))^*$ and that C is an I/O-automaton. The requirements on $\mathcal{F}$ follow directly from the definition of $\mathcal{F}$.

The conclusion of the lemma follows from the definition of N. If $t \in T(C)$ we can for any D construct a computation of $\Phi[D]$ with trace t by first performing a transition of form (3), whereafter $\Phi[D]$ behaves like C. If $t \in \lceil P(C) \rceil$ we can for any D with $t \in T(D)$ construct a computation of $\Phi[D]$ with trace t by simultaneously performing the sequence t on the events $E(D)$ and performing the required sequence of transitions of form (2). Note that neither of the fairness sets F_{even}, F_{odd}, or F_{other} is continuously enabled in such a computation.

Next, consider a computation of $\Phi[D]$ for an arbitrary D. If a transition of form (3) is performed, then the trace of the computation will be a trace of C, since N behaves as C after the transition of form (3). If no transition of form (3) is performed, then due to the fairness sets the sequence performed on $E(A)$ must be t. In this case any trace of the computation is possible.

The conclusion is that the sequence of events on $E(C)$ in a computation of $\Phi[D]$ can be any trace in $T(C)$ and can be any trace in $\lceil P(C) \rceil$ iff t is a trace of D. The conclusion of the lemma follows.

The case in which the sets $I(A)$, $O(A)$, $I(C)$, and $O(C)$ are not pairwise disjoint is treated in the same way as in the proof of Lemma 4.4. □

Theorem 4.7 *The relation* $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_T$ *is a minimal proper inclusion.* □

Proof: The theorem follows from Lemma 4.6 and Lemma 4.3, using $h(C) = CHAOS(C)$ and the observation that whenever $[\![A]\!]_P = [\![B]\!]_P$ and $[\![A]\!]_T \neq [\![B]\!]_T$ there is a finite or infinite sequence of events $t \in (E(A))^\dagger$ such that $t \notin T(A)$ but $t \in T(B)$ or vice versa. □

4.4 Proof that $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_Q$, $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_{PQ}$, and $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TQ}$ are minimal proper inclusions

The proofs that $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_Q$, $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_{PQ}$, and $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TQ}$ are minimal proper inclusions all use the same context. We therefore first present a lemma which presents this contexts and its essential properties.

Let I and O be disjoint sets of communication events, and let $t \in (I \cup O)^*$ be a finite sequence of communication events in $I \cup O$. Let $CLOCK_{I,O}(t)$ be an I/O-automaton with input events I and output events O which in parallel performs internal events. The internal events are stopped only if the sequence of events in I and O is t. The internal events are again resumed is the sequence of events on I and O are extended beyond t.

A more detailed description of $CLOCK_{I,O}(t)$ can be given as the I/O-automaton $\langle I, O, S, S^0, T, \mathcal{F}\rangle$, where

S is the set of finite sequences of events in $I \cup O$,

$S^0 = \langle\rangle$,

T contains the transitions

(1) $t_A \xrightarrow{e_A} t_A \bullet e_A$ for each $e_A \in (I \cup O)$ and each $t_A \in S$, and

(2) $t_A \xrightarrow{\tau} t_A$, for each $t_A \neq t$,

$\mathcal{F} = \{F\}$, where F contains all transitions of form (2).

Lemma 4.8 *Let A and C be I/O-automata. Let $\Phi_{C,t}[\cdot]$ be the context $C\|((\cdot\|CLOCK_{O(\cdot),I(\cdot)}(t))\backslash E(\cdot))$. Then the I/O-automaton $\Phi_{C,t}[A] = C\|((A\|CLOCK_{O(A),I(A)}(t))\backslash E(A))$ has the following properties.*

(1) $E(\Phi_{C,t}[A]) = E(C)$, $P(\Phi_{C,t}[A]) = P(C)$, and $T(\Phi_{C,t}[A]) = T(C)$,

(2) if $t \notin Q(A)$ then $Q(\Phi_{C,t}[A]) = \emptyset$,

(3) if $t \in Q(A)$ then $Q(\Phi_{C,t}[A]) = Q(C)$.

□

Proof: Property (1) follows by observing that $\Phi_{C,t}[A]$ is the parallel composition of C and an I/O-automaton without communication events (since the events $E(A)$ are abstracted away). Property (2) follows by the observation that if $t \notin Q(A)$ then $A\|CLOCK_{O(A),I(A)}(t)$ will perform infinitely many internal events in any computation. Property (3) follows by the observation that if $t \in Q(A)$ then there is a computation of $A\|CLOCK_{O(A),I(A)}(t)$ with trace t where both A and $CLOCK_{O(A),I(A)}(t)$ perform only finitely many internal events. In parallel with such a computation, C can perform any of its terminated computations. □

Using Lemma 4.8, we can now prove that $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_Q$, $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_{PQ}$, and $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TQ}$ are minimal proper inclusions. Let $TAUS$ be the I/O-automaton without communication events which in each computation performs an infinite sequence of internal events. $TAUS$ can be constructed with a single state, a single transition labeled by τ and a fairness set containing that transition. For an I/O-automaton A, let $DIV(A) = A\|TAUS$, i.e., $DIV(A)$ is an I/O-automaton which has the same traces as A and no terminated traces. We have $[\![DIV(A)]\!]_{TQ} = \langle I(A), O(A), T(A), \emptyset\rangle$.

Theorem 4.9 *The relation $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_Q$ is a minimal proper inclusion.* □

Proof: Assume that A and B satisfy $[\![A]\!]_E = [\![B]\!]_E$ and $[\![A]\!]_Q \neq [\![B]\!]_Q$. Without loss of generality, we can then assume that there is a finite sequence of events $t \in (E(A))^*$ such that $t \notin Q(A)$ but $t \in Q(B)$. Let C be an arbitrary I/O-automaton. For the context $\Phi_{C,t}[\cdot]$ of Lemma 4.8 we then have $[\![\Phi_{C,t}[A]]\!]_Q = [\![DIV(C)]\!]_Q$ and $[\![\Phi_{C,t}[B]]\!]_Q = [\![C]\!]_Q$. The theorem now follows from Lemma 4.3, using $h(C) = DIV(C)$. □

Theorem 4.10 *The relation* $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_{PQ}$ *is a minimal proper inclusion.* □

Proof: Analogous to the proof of Theorem 4.9, using $[\![\cdot]\!]_P$ instead of $[\![\cdot]\!]_E$, and using $[\![\cdot]\!]_{PQ}$ instead of $[\![\cdot]\!]_Q$. □

Theorem 4.11 *The relation* $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TQ}$ *is a minimal proper inclusion.* □

Proof: Analogous to the proof of Theorem 4.9 using $[\![\cdot]\!]_T$ instead of $[\![\cdot]\!]_E$, and using $[\![\cdot]\!]_{TQ}$ instead of $[\![\cdot]\!]_Q$. □

4.5 On the proper inclusions $[\![\cdot]\!]_Q \sqsubset [\![\cdot]\!]_{PQ}$ and $[\![\cdot]\!]_{PQ} \sqsubset [\![\cdot]\!]_{TQ}$

As stated in Section 4.1, the proper inclusions $[\![\cdot]\!]_Q \sqsubset [\![\cdot]\!]_{PQ}$ and $[\![\cdot]\!]_{PQ} \sqsubset [\![\cdot]\!]_{TQ}$ are not minimal. In this section, we shall show that there is exactly one compositional model between $[\![\cdot]\!]_Q$ and $[\![\cdot]\!]_{PQ}$ which we denote $[\![\cdot]\!]_{E/P\ Q}$, and exactly one compositional model between $[\![\cdot]\!]_{PQ}$ and $[\![\cdot]\!]_{TQ}$ which we denote $[\![\cdot]\!]_{P/T\ Q}$.

- The model $[\![\cdot]\!]_{E/P\ Q}$ is defined by letting the denotation $[\![A]\!]_{E/P\ Q}$ of an I/O-automaton A be $[\![A]\!]_Q$ if $Q(A) = \emptyset$ and be $[\![A]\!]_{PQ}$ if $Q(A) \neq \emptyset$.
- The model $[\![\cdot]\!]_{P/T\ Q}$ is defined by letting the denotation $[\![A]\!]_{P/T\ Q}$ of an I/O-automaton A be $[\![A]\!]_{PQ}$ if $Q(A) = \emptyset$ and be $[\![A]\!]_{TQ}$ if $Q(A) \neq \emptyset$.

Intuitively, the model $[\![\cdot]\!]_{E/P\ Q}$ is identical to the model $[\![\cdot]\!]_{PQ}$, except for those I/O-automata which have no terminated traces. For I/O-automata with no terminated traces $[\![\cdot]\!]_{E/P\ Q}$ is identical to $[\![\cdot]\!]_Q$. Similarly, the model $[\![\cdot]\!]_{P/T\ Q}$ is identical to the model $[\![\cdot]\!]_{TQ}$, except for those I/O-automata which have no terminated traces where $[\![\cdot]\!]_{P/T\ Q}$ is identical to $[\![\cdot]\!]_{PQ}$. The intuitive reason why $[\![\cdot]\!]_{E/P\ Q}$ is compositional is that the class of I/O-automata with no terminated traces is closed under the operations composition and abstraction. If one argument of an operation has no terminated traces, then $[\![\cdot]\!]_{E/P\ Q}$ gives less information about that argument. But this does not matter, since also the result of the operation has no terminated traces. The model $[\![\cdot]\!]_{P/T\ Q}$ is compositional for the same intuitive reason. We state this in the following theorem.

Theorem 4.12 $[\![\cdot]\!]_Q \sqsubset [\![\cdot]\!]_{E/P\ Q} \sqsubset [\![\cdot]\!]_{PQ}$ *and* $[\![\cdot]\!]_{PQ} \sqsubset [\![\cdot]\!]_{P/T\ Q} \sqsubset [\![\cdot]\!]_{TQ}$. *Furthermore, the models* $[\![\cdot]\!]_{E/P\ Q}$ *and* $[\![\cdot]\!]_{P/T\ Q}$ *are compositional.*

Proof: The first claim follows immediately from the definitions of $[\![\cdot]\!]_{E/P\ Q}$ and $[\![\cdot]\!]_{P/T\ Q}$. We shall prove that the composition and abstraction operations can be defined in the model $[\![\cdot]\!]_{E/P\ Q}$. If $Q(A_i) \neq \emptyset$ for $i = 1, \ldots, k$ then $[\![A_1 \| \cdots \| A_k]\!]_{E/P\ Q}$ can be obtained from $[\![A_1 \| \cdots \| A_k]\!]_{PQ}$ which can be obtained from $[\![A_1]\!]_{PQ}, \ldots, [\![A_k]\!]_{PQ}$ since $[\![\cdot]\!]_{PQ}$ is compositional. Since in this case $[\![A_i]\!]_{E/P\ Q}$ and $[\![A_i]\!]_{PQ}$ coincide, we conclude that $[\![A_1 \| \cdots \| A_k]\!]_{E/P\ Q}$ can be obtained from $[\![A_1]\!]_{E/P\ Q}, \ldots, [\![A_k]\!]_{E/P\ Q}$. Similarly, if $Q(A) \neq \emptyset$ then $[\![A \setminus E]\!]_{E/P\ Q}$ can be obtained from $[\![A]\!]_{PQ}$ which coincides with $[\![A]\!]_{E/P\ Q}$. If $Q(A_i) = \emptyset$ for some i between 1 and k then $Q(A_1 \| \cdots \| A_k) = \emptyset$. Hence, we only need the denotations $[\![A_1]\!]_Q, \ldots, [\![A_k]\!]_Q$ to obtain $[\![A_1 \| \cdots \| A_k]\!]_{E/P\ Q}$. Similarly, if $Q(A) = \emptyset$ then $Q(A \setminus E) = \emptyset$ and hence $[\![A \setminus E]\!]_{E/P\ Q}$ can be obtained from $[\![A]\!]_Q$. The proof that $[\![\cdot]\!]_{P/T\ Q}$ is compositional is analogous. □

The following two lemmas provide the basis for proving that $[\![\cdot]\!]_{E/P\ Q}$ is the only compositional model between $[\![\cdot]\!]_Q$ and $[\![\cdot]\!]_{PQ}$. Let *TAUS* be the I/O-automaton without communication events which in each computation performs an infinite sequence of internal events.

We thus have $[\![TAUS]\!]_{PQ} = \langle \emptyset, \emptyset, \{\langle\rangle\}, \emptyset \rangle$ and $[\![TAUS]\!]_{TQ} = \langle \emptyset, \emptyset, \{\langle\rangle\}, \emptyset \rangle$. Let $ALL1(C) = ALL(I(C), O(C))$. Let $ALL2(C)$ be an I/O-automaton, which nondeterministically either behaves like $ALL1(C) || TAUS$ or like C. The denotation of $ALL1(C) || TAUS$ in the model $[\![\cdot]\!]_{PQ}$ is $\langle I(C), O(C), (E(C))^*, \emptyset \rangle$. Thus the denotation $[\![ALL2(C)]\!]_{PQ}$ is $\langle I(C), O(C), (E(C))^*, Q(C) \rangle$ since it is the union of the denotations of $ALL1(C) || TAUS$ and of C.

Lemma 4.13 *Let A, B, and C be I/O-automata such that $[\![A]\!]_Q = [\![B]\!]_Q$ and $Q(A) \neq \emptyset$. Assume that there is a finite sequence of events $t \in E(A)^*$ such that $t \notin P(A)$ but $t \in P(B)$. Then there is a context $\Phi[\cdot]$ such that $[\![\Phi[A]]\!]_{PQ} = [\![C]\!]_{PQ}$ and $[\![\Phi[B]]\!]_{PQ} = [\![(ALL2(C)]\!]_{PQ}$.* □

Proof: Assume that $E(A)$ and $E(C)$ are disjoint. Let N be the I/O-automaton in the context $(\cdot || N) \setminus E(A)$ in the proof of Lemma 4.4. Let $TERM(A)$ be an I/O-automaton with $I(TERM(A)) = O(A)$ and $O(TERM(A)) = I(A)$ and $Q(TERM(A)) \cap Q(A) \neq \emptyset$. Thus $TERM(A)$ communicates with A and $(TERM(A) || A) \setminus E(A)$ has $\langle\rangle$ as a terminated trace. Let M be an I/O-automaton which nondeterministically behaves either like $N || TAUS$ or like like $C || TERM(A)$. We claim that the context $\Phi[\cdot] = (\cdot || M) \setminus E(A)$ satisfies the conditions of the lemma. By Lemma 4.4, noting that the inclusion of $TAUS$ removes all terminated traces,

$$[\![(N || TAUS || A) \setminus E(A)]\!]_{PQ} = \langle I(C), O(C), P(C), \emptyset \rangle \quad \text{and}$$
$$[\![(N || TAUS || B) \setminus E(A)]\!]_{PQ} = \langle I(C), O(C), P(ALL2(C)), \emptyset \rangle \ .$$

Since $(TERM(A) || A) \setminus E(A)$ has $\langle\rangle$ as a terminated trace, and only $\langle\rangle$ as a partial trace, we get

$$[\![(C || TERM(A) || A) \setminus E(A)]\!]_{PQ} = \langle I(C), O(C), P(C), Q(C) \rangle \quad \text{and}$$
$$[\![(C || TERM(A) || B) \setminus E(A)]\!]_{PQ} = \langle I(C), O(C), P(C), Q(C) \rangle$$

We conclude that

$$[\![(M || A) \setminus E(A)]\!]_{PQ} = \langle I(C), O(C), P(C), Q(C) \rangle \quad \text{and}$$
$$[\![(M || B) \setminus E(A)]\!]_{PQ} = \langle I(C), O(C), P(ALL2(C)), Q(C) \rangle$$

which gives the conclusion of the lemma.

The case in which the sets $E(A)$ and $E(C)$ are not disjoint is treated in the same way as in the proof of Lemma 4.4. □

Lemma 4.14 *Let A, B, and C be I/O-automata such that $[\![A]\!]_Q = [\![B]\!]_Q$ and $Q(A) = Q(B) = Q(C) = \emptyset$. Assume that there is a finite sequence of events $t \in E(A)^*$ such that $t \notin P(A)$ but $t \in P(B)$. Then there is a context $\Phi[\cdot]$, such that $[\![\Phi[A]]\!]_{PQ} = [\![C]\!]_{PQ}$ and $[\![\Phi[B]]\!]_{PQ} = [\![(ALL2(C)]\!]_{PQ}$.* □

Proof: The same as the proof of Lemma 4.4. Note that when the context is applied to either A or B then the result has no terminated traces, since A and B have no terminated traces. □

Theorem 4.15 *If the model $[\![\cdot]\!]$ is compositional and satisfies $[\![\cdot]\!]_Q \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_{PQ}$ then $[\![\cdot]\!] \cong [\![\cdot]\!]_{E/P\ Q}$.* □

Proof: First consider the I/O-automata which have at least one terminated trace. Lemma 4.13 and Lemma 4.3 using $h(C) = ALL2(C)$ show that for these I/O-automata there is no compositional model $[\![\cdot]\!]$ which satisfies $[\![\cdot]\!]_Q \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_{PQ}$. Using Lemma 4.14, we can prove the same

if we only consider I/O-automata with no terminated traces. Consequently, any compositional model $[\![\cdot]\!]$ of I/O-automata which satisfies $[\![\cdot]\!]_Q \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_{PQ}$ must combine one of $[\![\cdot]\!]_Q$ or $[\![\cdot]\!]_{PQ}$ for I/O-automata with some terminated traces with one of $[\![\cdot]\!]_Q$ or $[\![\cdot]\!]_{PQ}$ for I/O-automata with no terminated traces. There are two such combinations apart from the models $[\![\cdot]\!]_Q$ and $[\![\cdot]\!]_{PQ}$, and it is easy to check that $[\![\cdot]\!]_{E/P\ Q}$ is the only of these which is compositional. □

The following two lemmas provide the basis for proving that $[\![\cdot]\!]_{P/T\ Q}$ is the only compositional model between $[\![\cdot]\!]_{PQ}$ and $[\![\cdot]\!]_{TQ}$. Let $CHAOS2(C)$ be an I/O-automaton, which nondeterministically either behaves like $CHAOS(C)||TAUS$ or like C. The denotation of $CHAOS(C)||TAUS$ in the model $[\![\cdot]\!]_{TQ}$ is $\langle I(C), O(C), \lceil P(C) \rceil, \emptyset \rangle$. Thus the denotation of $CHAOS2(C)$ in the model $[\![\cdot]\!]_{TQ}$ is obtained as the union of the denotations of $CHAOS(C)||TAUS$ and of C and is $\langle I(C), O(C), \lceil P(C) \rceil, Q(C) \rangle$.

Lemma 4.16 *Let A, B, and C be I/O-automata such that $[\![A]\!]_{PQ} = [\![B]\!]_{PQ}$ and $Q(A) \neq \emptyset$. Assume that there is a finite or infinite sequence of events $t \in E(A)^\dagger$ such that $t \notin T(A)$ but $t \in T(B)$. Then there is a context $\Phi[\cdot]$, such that $[\![\Phi[A]]\!]_{TQ} = [\![C]\!]_{TQ}$ and $[\![\Phi[B]]\!]_{TQ} = [\![(CHAOS2(C)]\!]_{TQ}$.* □

Proof: Analogous to the proof of Lemma 4.13, using $CHAOS2(C)$ instead of $ALL2(C)$, and using the I/O-automaton N from Lemma 4.6. □

Lemma 4.17 *Let A, B, and C be I/O-automata such that $[\![A]\!]_{PQ} = [\![B]\!]_{PQ}$ and $Q(A) = Q(C) = \emptyset$. Assume that there is a finite or infinite sequence of events $t \in E(A)^\dagger$ such that $t \notin T(A)$ but $t \in T(B)$. Then there is a context $\Phi[\cdot]$, such that $[\![\Phi[A]]\!]_{TQ} = [\![C]\!]_{TQ}$ and $[\![\Phi[B]]\!]_{TQ} = [\![(CHAOS2(C)]\!]_{TQ}$.* □

Proof: The same as the proof of Lemma 4.6. Note that when the context is applied to either A or B then the result has no terminated traces, since A and B have no terminated traces. □

Theorem 4.18 *If the model $[\![\cdot]\!]$ is compositional and satisfies $[\![\cdot]\!]_{PQ} \sqsubset [\![\cdot]\!] \sqsubset [\![\cdot]\!]_{TQ}$ then $[\![\cdot]\!] = [\![\cdot]\!]_{P/T\ Q}$.* □

Proof: Analogous to the proof of Theorem 4.15. □

4.6 On the Proper Inclusion $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TD}$

Regarding divergence, the relation $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TD}$ is, as stated in Section 4.1, not minimal. In fact, there are several compositional models between $[\![\cdot]\!]_T$ and $[\![\cdot]\!]_{TD}$. We have not been able to characterize fully the models between $[\![\cdot]\!]_T$ and $[\![\cdot]\!]_{TD}$. In this section, we will present a few compositional models between $[\![\cdot]\!]_T$ and $[\![\cdot]\!]_{TD}$. For one of these models, called $[\![\cdot]\!]_{TI}$, the relation $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TI}$ is a minimal proper inclusion.

Let A be an I/O-automaton. An *environment* of A is an I/O-automaton H with $I(H) = O(A)$ and $O(H) = I(A)$. For an I/O-automaton A, define $IS(A)$ to be the set of finite sequences t in $T(A)$ for which there is an environment H of A such that each computation of $H||A$ is finite and $t \in T(H||A)$. Intuitively, if $t \in IS(A)$ then A can perform the trace t. Furthermore, the environment of A can enforce that the resulting computation terminates whenever the partial trace is still a prefix of t. Define the model $[\![\cdot]\!]_{TI}$ by $[\![A]\!]_{TI} = \langle I(A), O(A), T(A), IS(A) \rangle$.

Intuitively, $[\![A]\!]_{TI}$ gives information about which traces can be reached in a finite computation where the environment can at any point enforce termination.

As an example, consider the unbounded FIFO buffer of Example 2.3. Let BUF_{div} be the FIFO buffer of Example 2.3 with the difference that there is a partial trace t_{div} of the buffer such that when the sequence of communication events in a computation is t_{div} then BUF_{div} can nondeterministically decide either to continue to behave like a normal buffer or to start outputting an infinite sequence of messages regardless of future input. Intuitively, the trace t_{div} could be thought of as representing conditions under which the buffer may possibly malfunction and start producing messages continuously. Let t' be the shortest prefix of t_{div} that contains the same sequence of input events as t_{div}. Then $IS(BUF_{div})$ is the set of traces that do not have t' as a prefix. To see why, note that if t' is not a prefix of the current partial trace, than the environment can always enforce that the resulting computation be finite by ceasing to supply BUF_{div} with input messages. After having output all received messages, BUF_{div} will terminate. However, if t' is the current partial trace, then BUF_{div} can extend the partial trace to t_{div} by producing more output and thereafter cause an infinite computation.

Theorem 4.19 $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TI} \sqsubset [\![\cdot]\!]_{TD}$. □

Proof: The first relation is obvious. To check the second we verify that $[\![\cdot]\!]_{TI}$ can be obtained from $[\![\cdot]\!]_{TD}$. Given the denotation $[\![A]\!]_{TD}$, we can for each environment H of A check whether $H||A$ has no infinite or divergent traces. Thus we can check whether $t \in IS(A)$ by checking whether there is such a H with $t \in T(H||A)$. Consequently, $[\![\cdot]\!]_{TI}$ is uniquely determined by $[\![\cdot]\!]_{TD}$. □

Theorem 4.20 *The model* $[\![\cdot]\!]_{TI}$ *is compositional.* □

Proof: Assume that $A_1, \ldots, A_k$ are compatible I/O-automata and $A = A_1||\cdots||A_k$. We claim that $t \in IS(A)$ if and only if there is an environment H of A such that all traces u of $H||A$ satisfy

$$(\forall i) \qquad [u\lceil_{E(A_i)} \in IS(A_i)]$$

and $t \in T(H||A)$. The "if" direction is immediate, since no computations with trace in $IS(A_i)$ are divergent or infinite. To check the "only if", assume that $t \in IS(A)$, i.e., there is an environment H for A such that all traces of $H||A$ are finite and $t \in T(H||A)$. We then note that $H_i = H||A_1||\cdots||A_{i-1}||A_{i+1}||\cdots||A_k$ is an environment for A_i such that all computations of $H_i||A_i$ are finite. This means that all traces u of $H||A$ satisfy $u\lceil_{E(A_i)} \in IS(A_i)$. One can analogously prove that also the abstraction operation can be defined. □

Lemma 4.21 *Let* A, B, *and* C *be I/O-automata such that* $[\![A]\!]_T = [\![B]\!]_T$. *Assume that there is a finite sequence of events* $t \in E(A)^\dagger$ *such that* $t \notin IS(A)$ *but* $t \in IS(B)$. *Then there is a context* $\Phi[\cdot]$, *such that* $[\![\Phi[A]]\!]_T = [\![DIV(C)]\!]_T$ *and* $[\![\Phi[B]]\!]_T = [\![C]\!]_T$. □

Proof: Let H be an environment for B such that all computations of $H||B$ are finite and $t \in T(H||B)$. We claim that the lemma is satisfied by the context $\Phi[\cdot]C||((\cdot||H) \setminus E(A))$. To see the claim, note first that a computation of $\Phi[B]$ is infinite if and only if the corresponding computation of C is infinite, since $(B||H) \setminus E(A)$ will always perform a finite computation. Then note that $\Phi[A]$ has a divergent computation for each finite trace of C, since $(A||H) \setminus E(A)$ can perform an infinite sequence of internal transitions (otherwise we would have $t \in IS(A)$) regardless of the behavior of C. □

Theorem 4.22 *The relation* $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TI}$ *is a minimal proper inclusion.* □

Proof: Follows from Lemma 4.21 and Lemma 4.3 using $h(C) = DIV(C)$. □

In this section, we shall also define another compositional model between $[\![\cdot]\!]_T$ and $[\![\cdot]\!]_{TD}$, to show that there is more than one such compositional model. Define $S(A)$ to be the set of finite sequences t in $P(A)$ for which there is an environment H of A such that $t \in P(H||A)$ and each computation of $H||A$, whose trace has t as a prefix, is finite. Intuitively, if $t \in S(A)$ then the environment A can enforce that the resulting computation terminates if the computation has reached a point where the sequence of events has become t (we have used the letter S to stand for "Stoppable" traces). Note that, in contrast with sequences in $IS(A)$, it may be the case that computations with traces that are prefixes of t can be divergent without the environment being able to enforce a finite computation. Define the model $[\![\cdot]\!]_{TSD}$ by $[\![A]\!]_{TSD} = \langle I(A), O(A), T(A), S(A), S(A) \cap D(A) \rangle$.

Theorem 4.23 $[\![\cdot]\!]_T \sqsubset [\![\cdot]\!]_{TSD} \sqsubset [\![\cdot]\!]_{TD}$. □

Proof: Analogous with the proof of Theorem 4.19. □

Theorem 4.24 *The model* $[\![\cdot]\!]_{TSD}$ *is compositional.* □

Proof: Similar to the proof of Theorem 4.20. □

5 Applications to Full Abstraction Results

Full abstraction is an important property of models, which means that a model in an optimal way combines abstraction from irrelevant detail with compositionality. Intuitively, a model 2 is fully abstract with respect to a model 1 if model 2 has added precisely enough information to model 1 for attaining compositionality. In this section, we indicate how our hierarchy can be applied to obtain results about full abstraction. The main idea is that given a non-compositional model in our hierarchy, one can obtain a fully abstract model by going up in the hierarchy until reaching a compositional model.

We begin with some standard definitions.

Definition 5.1 *Let* $[\![\cdot]\!]_{\mathcal{D}}$ *and* $[\![\cdot]\!]_{\mathcal{O}}$ *be two models (of I/O-automata). The model* $[\![\cdot]\!]_{\mathcal{D}}$ *is said to be* fully abstract *with respect to the model* $[\![\cdot]\!]_{\mathcal{O}}$ *if* $[\![\cdot]\!]_{\mathcal{O}} \sqsubseteq [\![\cdot]\!]_{\mathcal{D}}$ *and for any compositional model* $[\![\cdot]\!]$ *it is the case that* $[\![\cdot]\!]_{\mathcal{O}} \sqsubseteq [\![\cdot]\!]$ *implies* $[\![\cdot]\!]_{\mathcal{D}} \sqsubseteq [\![\cdot]\!]$.

Intuitively, $[\![\cdot]\!]_{\mathcal{D}}$ is the minimal compositional model which contains at least as much information as $[\![\cdot]\!]_{\mathcal{O}}$. The following proposition shows that a fully abstract model always exists and characterizes a fully abstract model.

Proposition 5.2 *Let* $[\![\cdot]\!]_{\mathcal{O}}$ *be a model. There is a unique model (up to* $\cong$*)* $[\![\cdot]\!]_{\mathcal{D}}$ *which is fully abstract with respect to* $[\![\cdot]\!]_{\mathcal{O}}$*. The model* $[\![\cdot]\!]_{\mathcal{D}}$ *is characterized by*

$$\text{for all } A_1 \text{ and } A_2 \qquad \Big[[\![A_1]\!]_{\mathcal{D}} = [\![A_2]\!]_{\mathcal{D}} \iff (\forall \text{ contexts } \Phi[\cdot])\ [\![\Phi[A_1]]\!]_{\mathcal{O}} = [\![\Phi[A_2]]\!]_{\mathcal{O}} \Big]$$

Proof: We shall show that $[\![\cdot]\!]_{\mathcal{D}}$ exists and satisfies the characterization. This will imply uniqueness of $[\![\cdot]\!]_{\mathcal{D}}$. If $[\![\cdot]\!]_{\mathcal{D}}$ is compositional and $[\![\cdot]\!]_{\mathcal{O}} \sqsubseteq [\![\cdot]\!]_{\mathcal{D}}$ then the implication $[\![A_1]\!]_{\mathcal{D}} = [\![A_2]\!]_{\mathcal{D}} \Longrightarrow (\forall \Phi[\cdot])[[\![\Phi[A_1]]\!]_{\mathcal{O}} = [\![\Phi[A_2]]\!]_{\mathcal{O}}]$ holds for all A_1, A_2. We thus have a restriction on which I/O-automata can be identified by $[\![\cdot]\!]_{\mathcal{D}}$. The minimal model satisfying these restrictions is the model defined by the characterization of the proposition. We must only check that this model is compositional. But compositionality follows from the observation that for a given context $\Phi_0[\cdot]$

$$\begin{array}{lll} [\![A_1]\!]_{\mathcal{D}} = [\![A_2]\!]_{\mathcal{D}} & \Longrightarrow & (\forall \Phi[\cdot])[[\![\Phi[A_1]]\!]_{\mathcal{O}} = [\![\Phi[A_2]]\!]_{\mathcal{O}}] \quad \Longrightarrow \\ (\forall \Phi[\cdot])[[\![\Phi[\Phi_0[A_1]]]\!]_{\mathcal{O}} = [\![\Phi[\Phi_0[A_2]]]\!]_{\mathcal{O}}] & \Longrightarrow & [[\![\Phi_0[A_1]]\!]_{\mathcal{D}} = [\![\Phi_0[A_2]]\!]_{\mathcal{D}}] \end{array}$$

□

Proposition 5.3 *Assume that* $[\![\cdot]\!]_1 \sqsubseteq [\![\cdot]\!]_2$, *and that there is no compositional model* $[\![\cdot]\!]$ *with* $[\![\cdot]\!]_1 \sqsubseteq [\![\cdot]\!] \sqsubset [\![\cdot]\!]_2$. *If* $[\![\cdot]\!]_2$ *is compositional then* $[\![\cdot]\!]_2$ *is fully abstract with respect to* $[\![\cdot]\!]_1$.

Proof: Follows from the existence and uniqueness of the fully abstract model: if $[\![\cdot]\!]_2$ is not fully abstract with respect to $[\![\cdot]\!]_1$ then there is a model $[\![\cdot]\!]$ with $[\![\cdot]\!]_1 \sqsubseteq [\![\cdot]\!] \sqsubset [\![\cdot]\!]_2$. □

Proposition 5.4 *Assume that* $[\![\cdot]\!]_{\mathcal{D}}$ *is fully abstract with respect to* $[\![\cdot]\!]_{\mathcal{O}}$, *that* $[\![\cdot]\!]'_{\mathcal{D}}$ *is fully abstract with respect to* $[\![\cdot]\!]'_{\mathcal{O}}$ *and that* $[\![\cdot]\!]_{\mathcal{O}} \sqsubseteq [\![\cdot]\!]'_{\mathcal{O}}$. *Then* $[\![\cdot]\!]_{\mathcal{D}} \sqsubseteq [\![\cdot]\!]'_{\mathcal{D}}$.

Proof: Let A_1 and A_2 be arbitrary I/O-automata. By Proposition 5.2 and $[\![\cdot]\!]_{\mathcal{O}} \sqsubseteq [\![\cdot]\!]'_{\mathcal{O}}$. we have

$$\begin{array}{lll} [\![A_1]\!]'_{\mathcal{D}} = [\![A_2]\!]'_{\mathcal{D}} & \Longrightarrow & (\forall \Phi[\cdot])[\![\Phi[A_1]]\!]'_{\mathcal{O}} = [\![\Phi[A_2]]\!]'_{\mathcal{O}} \quad \Longrightarrow \\ (\forall \Phi[\cdot])[\![\Phi[A_1]]\!]_{\mathcal{O}} = [\![\Phi[A_2]]\!]_{\mathcal{O}} & \Longrightarrow & [\![A_1]\!]_{\mathcal{D}} = [\![A_2]\!]_{\mathcal{D}} \end{array}$$

which implies $[\![\cdot]\!]_{\mathcal{D}} \sqsubseteq [\![\cdot]\!]'_{\mathcal{D}}$. □

Using the two last propositions, we can now apply the hierarchy to obtain various full abstraction results. We shall illustrate this by one particular such result.

Let C be a computation of the I/O-automaton A. Define the *counter mapping* of C as a mapping m from $I(A) \cup O(A)$ to natural numbers or ∞ which for each communication event e gives the number of occurrences of e in that computation. Let $M(A)$ be the set of counter mappings of computations of A. Define the *counter model* $[\![\cdot]\!]_M$ of I/O-automata by letting the denotation $[\![A]\!]_M$ be the tuple $\langle I(A), O(A), M(A)\rangle$. The counter model is related to the history model of dataflow networks, which was originally defined by Kahn [Kah74]. For nondeterministic dataflow networks, the model $[\![\cdot]\!]_T$ is fully abstract with respect to the history model [Jon89]. Several other works present related full abstraction results [RT89, Kok87, Rus89]. We shall now show that for I/O-automata, we can prove an analogous result for the counter model as a direct application of the hierarchy in Figure 1 and the previous propositions.

Let $[\![\cdot]\!]_{\mathcal{D}}$ be a sought model which is fully abstract with respect to $[\![\cdot]\!]_M$. Let e be a communication event. Let $[\![\cdot]\!]_m$ be a model such that $[\![A]\!]_m$ in addition to $I(A)$ and $O(A)$ gives the possible numbers of occurrences of e in partial computations of A. Thus, $[\![\cdot]\!]_m$ relates to $[\![\cdot]\!]_M$ in an analogous way as $[\![\cdot]\!]_P$ relates to $[\![\cdot]\!]_T$. Clearly $[\![\cdot]\!]_E \sqsubset [\![\cdot]\!]_m \sqsubset [\![\cdot]\!]_P$ and $[\![\cdot]\!]_m \sqsubset [\![\cdot]\!]_M \sqsubset [\![\cdot]\!]_T$ whence by Propositions 5.3 and 5.4 and Theorem 4.5 we get $[\![\cdot]\!]_P \sqsubseteq [\![\cdot]\!]_{\mathcal{D}} \sqsubseteq [\![\cdot]\!]_T$. Since $[\![\cdot]\!]_M \not\sqsubseteq [\![\cdot]\!]_P$ we even get $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_{\mathcal{D}} \sqsubseteq [\![\cdot]\!]_T$. Now the fact that $[\![\cdot]\!]_P \sqsubset [\![\cdot]\!]_T$ is a minimal proper inclusion (Theorem 4.7) implies that $[\![\cdot]\!]_{\mathcal{D}} \cong [\![\cdot]\!]_T$.

6 Conclusions

We have presented a variety of compositional models for I/O-automata. All our models are based on various forms of traces. For different applications, one may want to use different models. We have organized our models into a hierarchy where the more complex and informative models are higher up in the hierarchy. The main contribution of the paper is to investigate parts of our hierarchy which is between the models that we have defined. Interestingly enough, it turns out that in many cases the part of the hierarchy which is between two adjacent models does not contain any compositional model.

Our work shows is that the requirement of compositionality for a model can impose rather strong constraints. In the framework which we have considered the concepts of safety as represented by prefix-closed sets of finite traces, liveness as represented by completed traces, and termination as represented by terminated traces, represent "indivisible units of description", in the sense that there is no model which can describe a part of the information provided by e.g. the some of terminated traces, in a compositional way. For divergence, the situation is more complicated. It is here interesting to note that the concept of divergence is problematic also in the work on semantics for CSP and CCS, and many solutions for the treatment of divergence have been proposed (e.g. [BR85, Wal88]).

Our work can also be applied to derive results about full abstraction for semantic models. Assuming a model which represents properties that are relevant for a particular problem at hand, one can obtain a fully abstract model by going upwards in the hierarchy until one reaches a compositional model.

In this paper, we have used a set-theoretical definition of I/O-automata, which allows I/O-automata with uncomputable sets of states and transitions. The results that only involve finite traces (i.e., the results that concern only partial traces and terminated traces) can be accommodated to suit a definition of I/O-automata which requires the sets of states, transitions, etc. to be computable. However, for the results about models that represent liveness and fairness we do not know how to restrict the results to "computable" I/O-automata. This is partly due to the fact that the concepts of fairness and liveness are in some respects uncomputable.

Our results about absence of compositional models are stated for the class of I/O-automata without any further restrictions. In many applications, e.g. to asynchronously communicating systems, one considers I/O-automata which satisfy some extra restrictions. One such extra restriction could be that an input event must not immediately affect the entire behavior of the I/O-automaton, but that input events can e.g. enable or disable output events only after some delay. In the paper [JK91] with Joost Kok, we investigate whether our results still hold for nondeterministic dataflow networks. Our preliminary investigations show that most results about the absence of compositional models for I/O-automata indeed carry over to dataflow networks.

Hierarchies of semantic models of communicating systems have also been defined and studied by Olderog and Hoare [OH86], by Reed and Roscoe [Ree90, RR86, RR88], and by van Glabbeek and Vaandrager [vGV87]. These works develop semantic models for communicating systems and organize them into some form of hierarchy. In contrast to our work, the above works do not prove the absence of compositional models in the hierarchy.

Olderog and Hoare [OH86] develop several semantic models for CSP where communication is synchronous. The semantics of a process is defined as a set of observations (our traces correspond to such observations). Several models are defined, depending on how detailed the

observations are. Reed and Roscoe [Ree90] present a rather elaborate hierarchy of models for timed CSP. The models differ in whether they represent timing or deadlock information and in their relation to the concept of stability. One purpose of the hierarchy of Reed and Roscoe is to be able to reason about a particular problem (e.g. verifying a property of a system) in the model which represents only the information relevant for that problem. Projection mappings between the models then allow to translate the results to other models in the hierarchy. It would be interesting to see whether gaps between models in these hierarchies for CSP are empty of models with desirable properties or not. Van Glabbeek and Vaandrager [vGV87] study several models of Petri nets, based on bisimulation. One purpose is to study the relation between true concurrency and interleaving models.

Acknowledgment

The author is grateful to Joost Kok and Joachim Parrow for fruitful discussions and comments.

References

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.

[BM85] R.J.R. Back and H. Mannila. On the suitability of trace semantics for modular proofs of communicating processes. *Theoretical Computer Science*, 39(1):47–68, 1985.

[BR85] S.D. Brookes and A.W. Roscoe. An improved failures model for communicating processes. In Brookes, Roscoe, and Winskel, editors, *Proc. Seminar on Concurrency, 1984*, volume 197 of *Lecture Notes in Computer Science*, pages 268–280. Springer Verlag, 1985.

[dNH84] R. de Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[Fra86] N. Francez. *Fairness.* Springer Verlag, 1986.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[JK91] B. Jonsson and J. Kok. A hierarchy of compositional dataflow models, Jan. 1991. In Progress.

[Jon85] B. Jonsson. A model and proof system for asynchronous networks. In *Proc.* 4^{th} *ACM Symp. on Principles of Distributed Computing, Minaki, Canada*, pages 49–58, Minaki, Canada, 1985.

[Jon87] B. Jonsson. *Compositional Verification of Distributed Systems.* PhD thesis, Dept. of Computer Systems, Uppsala University, Sweden, Uppsala, Sweden, 1987. Available as report DoCS 87/09.

[Jon89] B. Jonsson. A fully abstract trace model for dataflow networks. In *Proc.* 16^{th} *ACM Symp. on Principles of Programming Languages*, pages 155–165, 1989.

[Jon90] B. Jonsson. On decomposing and refining specifications of distributed systems. In de Bakker, de Roever, and Rozenberg, editors, *Proc. REX Workshop on Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 361–385. Springer Verlag, 1990.

[Kah74] G. Kahn. The semantics of a simple language for parallel programming. In *IFIP 74*, pages 471–475. North-Holland, 1974.

[Kok87] J.N. Kok. A fully abstract semantics for data flow nets. In *Proc. PARLE, LNCS 259*, volume 259 of *Lecture Notes in Computer Science*, pages 351–368. Springer Verlag, 1987.

[LAJ81] Lehman, A.Pnueli, and J.Stavi. Impartiality, justice and fairness: The ethics of concurrent termination. In *Proc. ICALP '81*, volume 115 of *Lecture Notes in Computer Science*. Springer Verlag, 1981.

[LT87] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proc.* 6^{th} *ACM Symp. on Principles of Distributed Computing, Vancouver, Canada*, pages 137–151, 1987.

[MC81] J. Misra and K. M. Chandy. Proofs of networks of processes. *IEEE Trans. on Software Engineering*, SE-7(4):417–426, July 1981.

[MCS82] J. Misra, K. M. Chandy, and T. Smith. Proving safety and liveness of communicating processes with examples. In *Proc. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, pages 201–208, 1982.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[Mis84] J. Misra. Reasoning about networks of communicating processes. In *INRIA Advanced Nato Study Institute on Logics and Models for Verification and Specification of Concurrent Systems*, La Colle sur Loupe, France, 1984.

[MP81] Z. Manna and A. Pnueli. The temporal framework for concurrent programs. In Boyer and Moore, editors, *The Correctness Problem in Computer Science*, pages 215–274. Academic Press, 1981.

[NDGO86] V. Nguyen, A. Demers, D. Gries, and S. Owicki. A model and temporal proof system for networks of processes. *Distributed Computing*, 1(1):7–25, 1986.

[OH86] E.R. Olderog and C.A.R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23(1):9–66, 1986.

[Ora89] F. Orava. Verifying safety and deadlock properties of networks of asynchronously communicating processes. In *Proc.* 9^{th} *IFIP WG6.1 Symp. on Protocol Specification, Testing, and Verification*, Twente, Holland, 1989.

[Plo81] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Denmark, 1981.

[Ree90] G.M. Reed. A hierarchy of domains for real-time distributed computing. In Main, Melton, Mislove, and Schmidt, editors, *Proc. 5th Int. Conf. on Mathematical Foundations of Programming Semantics, New Orleans, Louisiana*, volume 442 of *Lecture Notes in Computer Science*, pages 80–128. Springer Verlag, March 1990.

[RR86] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. In *Proc. ICALP '86*, volume 226 of *Lecture Notes in Computer Science*, pages 314–323. Springer Verlag, 1986.

[RR88] G.M. Reed and A.W. Roscoe. Metric spaces as models for real-time concurrency. In *Proc. 3rd Workshop on Math. Found. of Progr. Lang. Semantics*, volume 298 of *LNCS*. Springer Verlag, 1988.

[RT89] A. Rabinovich and B.A. Trakhtenbrot. Nets of processes and data flow. In de Bakker, de Roever, and Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 574–602. Springer Verlag, 1989.

[Rus89] J.R. Russell. Full abstraction for nondeterministic dataflow networks. In *Proc. 30th Annual Symp. Foundations of Computer Science*, 1989.

[Sta84] E.W. Stark. *Foundations of a Theory of Specification for Distributed Systems.* PhD thesis, Massachussetts Inst. of Technology, 1984. Available as Report No. MIT/LCS/TR-342.

[vGV87] R. J. van Glabbeek and F. W. Vaandrager. Petri net models for algebraic theories of concurency. In *Proc. PARLE*, volume 259 of *Lecture Notes in Computer Science*, pages 224–242. Springer Verlag, 1987.

[Wal88] D. Walker. Bisimulation and divergence. In *Proc. 3rd IEEE Int. Symp. on Logic in Computer Science*, pages 186–192, 1988.

[Zwi89] J. Zwiers. *Compositionality, Concurrency and Partial Correctness*, volume 321 of *Lecture Notes in Computer Science.* Springer Verlag, 1989.

A Proofs of some Theorems

Proof of Theorem 3.4

Let $A = [\![A_1 \| \ldots \| A_k]\!]$. We shall prove that

$$P(A) = \{t \in (E(A))^* \ : \ (\forall i) \ \ t\lceil_{E(A_i)} \in P(A_i)\}$$

$$T(A) = \{t \in (E(A))^\dagger \ : \ (\forall i) \ \ t\lceil_{E(A_i)} \in T(A_i)\}$$

$$Q(A) = \{t \in (E(A))^* \ : \ (\forall i) \ \ t\lceil_{E(A_i)} \in Q(A_i)\}$$

$$D(A) = \{t \in (E(A))^* \ : \ (\exists i) \ [t\lceil_{E(A_i)} \in D(A_i) \quad \wedge \quad (\forall j \neq i) \ \ t\lceil_{E(A_j)} \in (T(A_j) \cup D(A_j))]\}.$$

We first prove that if a trace t is in the set on the left-hand side, then t is in the set on the right-hand side of each equality. Consider a finite or infinite sequence

$$\mathcal{C} \quad = \quad s^0 \xrightarrow{e^1} s^1 \xrightarrow{e^2} \cdots \xrightarrow{e^n} s^n \xrightarrow{e^{n+1}} \cdots$$

of transitions of A where s^0 is an initial state of A. Let $\mathcal{C}_i$ be the projection of $\mathcal{C}$ onto A_i. More precisely, $\mathcal{C}_i$ is obtained by replacing each transition

$$\langle {s^{n-1}}_1, \ldots, {s^{n-1}}_k \rangle \xrightarrow{e^n} \langle {s^n}_1, \ldots, {s^n}_k \rangle$$

in $\mathcal{C}$ by $s_i^{n-1} \xrightarrow{e^n} s_i^n$ if $e^n \in (I(A) \cup O(A))$, by $s_i^{n-1} \xrightarrow{\tau} s_i^n$ if $s_i^{n-1} \xrightarrow{\tau} s_i^n$ is a transition of A_i, and otherwise by s_i^n. By Definition 2.4 the result is a sequence of transitions of A_i. Thus if $\mathcal{C}$ is a partial computation of A then $\mathcal{C}_i$ is a partial computation of A_i. Let t be the sequence of communication events in $\mathcal{C}$ and let t_i be the sequence of communication events in $\mathcal{C}_i$. It follows that $t_i = t\lceil_{E(A_i)}$. Furthemore, t_i is a partial trace if t is a partial trace.

We claim that $\mathcal{C}_i$ is a computation of A_i if $\mathcal{C}$ is a computation of A. We must then check that the fairness requirements of A_i are satisfied in $\mathcal{C}_i$. So assume that a fairness set F_i of A_i is enabled continuously in $\mathcal{C}_i$. The fairness set F_i induces a fairness set F of A according to Definition 2.4. By condition 2 of Definition 2.1 any transition $s_i \xrightarrow{e} s_i'$ in F_i is labelled either by τ or by an output event of A_i. Thus, whenever a transition in F_i is enabled in $\mathcal{C}_i$ a corresponding transition in F will be enabled in the corresponding state of $\mathcal{C}$, since by condition 1 of Definition 2.1 there is always a transition of A which is projected onto $s_i \xrightarrow{e} s_i'$. It follows that F is continuously enabled in $\mathcal{C}$. Therefore, a transition from F will be performed in $\mathcal{C}$, hence some transition from F_i will eventually be performed in $\mathcal{C}_i$. Thus $\mathcal{C}_i$ satisfies the fairness requirements of A_i, and hence $t_i = t\lceil_{E(A_i)}$ is a trace of A_i if t is a trace of i. If $\mathcal{C}$ is finite, then $\mathcal{C}_i$ is of course also finite. Hence t_i is a terminated trace if t is. Assume finally that t and hence $\mathcal{C}$ is divergent. Then t_i is certainly finite, and one of $\mathcal{C}_1, \ldots, \mathcal{C}_k$ must be infinite and divergent.

We next prove that if a trace t is in the set on the right-hand side, then t is in the set on the left-hand side of each equality. Assume that there is a sequence t such that for each i, the sequence $t_i = t\lceil_{E(A_i)}$ is the sequence of events in a computation or partial computation $\mathcal{C}_i$ of A_i. If $q = e^1\ e^2\ e^3\ \ldots$ then q_i is a subsequence $e^{i_1}\ e^{i_2}\ e^{i_3}\ \ldots$ of q. Then $\mathcal{C}_i$ can be written as

$$\gamma_i^0 \xrightarrow{e^{i_1}} \gamma_i^1 \xrightarrow{e^{i_2}} \cdots \xrightarrow{e^{i_m}} \gamma_i^m \xrightarrow{e^{i_{m+1}}} \cdots$$

where each γ_i^m is a sequence of τ-transitions of A_i.

Define an *extension* of a τ-transition $s_i \xrightarrow{\tau} s_i'$ of A_i to be a transition

$$\langle s_1, \ldots, s_i, \ldots, s_k \rangle \xrightarrow{\tau} \langle s_1, \ldots, s_i', \ldots, s_k \rangle$$

of A, which keeps all components except s_i unchanged. The term *extension* is extended to sequences of τ-transitions of A_i in the natural way.

For each $n = 0, 1, 2 \ldots$, construct the sequence γ^n of transitions of A by creating an extension of each γ_i^m for which $n+1$ is equal to i_{m+1}, and then (if there are several such extensions) concatenating these extensions. The first state of γ^{n+1} is obtained from the last state of γ_i^m by performing the transition between the last state of γ_i^{m+1} and the first state of γ_i^n whenever $n+1$ is equal to i_{m+1}, otherwise by preserving the last state of γ_i^{m+1}. The first state of γ^0 should consist of the initial states of each γ_i^0. We can now conclude from Definition 2.4 that

$$\mathcal{C} = \gamma^0 \xrightarrow{e^1} \gamma^1 \xrightarrow{e^2} \cdots \xrightarrow{e^n} \gamma^n \xrightarrow{e^{n+1}} \cdots$$

is a sequence of transitions of A Thus t is a partial trace if each t_i is a partial trace.

We claim that $\mathcal{C}$ is a computation of A if $\mathcal{C}_i$ is a computation of A_i for each i. We need to consider the fairness sets of A. Assume that a fairness set F of A is continuously enabled beyond some point in $\mathcal{C}$. There is a F_i such that F is obtained from the fairness set F_i of A_i. It follows that F_i is continuously enabled in $\mathcal{C}_i$. Hence some transition from F_i must be performed in $\mathcal{C}_i$, hence a transition from F is performed in $\mathcal{C}$. Hence t is a trace if each t_i is a trace. Since $\mathcal{C}$ is finite if each $\mathcal{C}_i$ is finite, we conclude that t is a terminated trace if each t_i is a terminated trace. Assume finally, that one $\mathcal{C}_i$ is infinite and all t_j are finite. Then t is finite, and $\mathcal{C}$ is finite, whence t is a divergent trace.

The rules for the abstraction operation are immediate from Definition 2.5. □