Acta Informatica 23, 9-66 (1986)

Specification-Oriented Semantics for Communicating Processes*

E.-R. Olderog[1] and C.A.R. Hoare[2]

[1] Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, D-2300 Kiel 1, Federal Republic of Germany

[2] Oxford University Computing Laboratory, Programming Research Group, Oxford OX1 3QD, UK

Summary. A process P satisfies a specification S if every observation we can make of the behaviour of P is allowed by S. We use this idea of process correctness as a starting point for developing a specific form of denotational semantics for processes, called here specification - oriented semantics. This approach serves as a uniform framework for generating and relating a series of increasingly sophisticated denotational models for Communicating Processes.

These models differ in the underlying structure of their observations which influences both the number of representable language operators and the induced notion of process correctness. Safety properties are treated by all models; the more sophisticated models also permit proofs of certain liveness properties. An important feature of the models is a special *hiding* operator which abstracts from internal process activity. This allows large processes to be composed hierarchically from networks of smaller ones in such a way that proofs of the whole are constructed from proofs of its components. We also show the consistency of our denotational models w.r.t. a simple operational semantics based on transitions which make internal process activity explicit.

Contents

* A preliminary version of this paper appeared in [42]

1. Introduction

For concurrent programs - even when restricted to a particular style like Communicating Processes - a variety of semantical models have been proposed (e.g. [7, 13, 34]). Each of these different models can be viewed as describing certain aspects of a complex behaviour of programs. It seems desirable to bring some order into these semantical models so that one will be able to recommend each model for the purposes and applications for which it is best suited.

This leads us to pursue the following aims in our paper:

(1) The semantics of concurrent programs should lead to a *simple correctness criterion*, and simple proofs of correctness.

(2) The semantics should *abstract from the internal activity* of concurrent programs in order to allow large programs to be composed hierarchically as networks of smaller ones.

(3) Systematic methods should be developed for *generating sound semantical models* for different purposes and applications.

(4) Existing semantic models should *related to each other* in a clear system of classification.

We concentrate here on an application to Communicating Processes and develop a general framework in which we pursue the aims (1)–(4). In different settings, steps towards some of these aims can also be found in recent work by [7, 8, 12, 15, 36, 37, 40]. Let us now outline the approach of our paper.

(0) The Language. Informally, Communicating Processes is a programming language for describing networks of processes which work in parallel and communicate with each other in a synchronised way [25]. But the emphasis is here on studying the fundamental concepts involved rather than presenting a full programming notation such as [30]. Our version of Communicating Processes includes the concepts of deadlock, divergence, communication, internal and external nondeterminism, parallel composition with synchronisation, hiding of communications, and recursion (Sect. 3).

(1) Correctness. A process P satisfies a specification S, abbreviated $P\ \mathsf{sat}\ S$, if every observation we can make of the behaviour of P is allowed by S. We use this idea of process correctness as a starting point for developing a specific

form of denotational semantics for processes, called here specification-oriented semantics.

We begin with a sets Obs of observations together with a simple algebraic structure and define specifications S as certain subsets of Obs reflecting this structure (Sect. 4). The idea is that a specification S describes a set of nondeterministic possibilities of observations. This suggests the following ordering $\sqsubseteq$ among specifications:

$$S_1 \sqsubseteq S_2 \quad \text{iff} \quad S_1 \supseteq S_2.$$

This is the Smyth-order originally introduced in the context of nondeterministic state transformers [48]: $S_1 \sqsubseteq S_2$ means that S_2 is more deterministic than S_1.

A specification-oriented semantics assigns denotationally to every process P a specification $\mathscr{M}[\![P]\!]$ such that P sat S is expressed by $S \sqsubseteq \mathscr{M}[\![P]\!]$, i.e. $\mathscr{M}[\![P]\!]$ is within the range of nondeterminism permitted by S (Sect. 5). A process P is therefore identified by the strongest specification which it satisfies. To this end, the set Spec of strongest specifications over Obs forms a complete partial order under $\sqsubseteq$ and the semantics $\mathscr{M}[\![\cdot]\!]$ maps every syntactic constructor of the programming language onto a $\sqsubseteq$-continuous operator on specifications. This enables us to treat recursion in the usual way.

(2) Abstraction. Abstraction is realised in two ways.

Firstly, the hiding operator of Communicating Processes turns the concept of abstraction into an explicit language construct. Informally, hiding localises all communications on internal network channels. This allows us to construct a larger process by first constructing its components, then connecting them as desired and finally hiding those connections which are regarded as internal. A simple example will illustrate this point (Sect. 6).

Secondly, observations themselves are not allowed to mention internal process activity. This idea is formalised by imposing – in addition to the algebraic structure already mentioned – also a certain logical structure on observations (Sect. 13).

(3) Generality. The algebraic structure of observations is used to derive two general constructions for $\sqsubseteq$-continuous operators on specifications which are typical for Communicating Processes (Sect. 5). The simplest way of defining such an operator is by pointwise application of a relationship g between observations, i.e. to consider

$$\mathscr{O}_g(S) = \{y \mid \exists x \in S : x\, g\, y\}.$$

But this operator can be proved continuous only for a restricted class of relations g. It turns out that most of the operators in Communicating Processes satisfy this restriction, but not the crucial hiding operators. These are more complicated because the possibility of *infinitely many* hidden communications has to be considered. We present an abstract analysis of such hiding relations g and show that the definition

$$C_g(S) = \mathscr{O}_g(S) \cup \mathscr{O}_g^{\infty}(S)$$

yields a continuous operator. Here $\mathcal{O}_g^\infty(S)$ is an auxiliary operator dealing with the possibility of divergence.

(4) Classification. In the main part of our paper we apply this specification-oriented approach to semantics to systematically generate and relate a series of increasingly sophisticated denotational models for Communicating Processes (Sects. 6–11). These models differ in the underlying structure of their observations; and this influences both the number of representable language operators and the notion of process correctness expressed by P sat S. This suggests that for each particular application the simplest adequate model should be chosen.

The simplest model is the *Counter Model* $\mathscr{C}$ which reflects the idea of separate channel histories [27, 28]. We show that $\mathscr{C}$ can deal adequately only with acyclic or tree-like networks of processes. Arbitrary networks require the *Trace Model* $\mathscr{T}$ instead [26, 37]. However, both $\mathscr{C}$ and $\mathscr{T}$ are unable to deal satisfactory with diverging processes. This requires a further refinement of our observations leading to the *Divergence Model* $\mathscr{D}$.

In $\mathscr{C}$, $\mathscr{T}$ and $\mathscr{D}$ only *safety properties* can be described by P sat S. Also the concept of external nondeterminism is not yet available. Dealing with the full language of Communicating Processes and with certain *liveness properties* calls for the *Readiness Model* $\mathscr{R}$ [18, 27]. We characterise the kind of liveness properties which are expressible in $\mathscr{R}$. By studying the algebraic laws of $\mathscr{R}$ we find that finite, i.e. non-recursive processes are not always reducible to processes without parallel composition and hiding. This shortcoming of $\mathscr{R}$ is avoided in the *Refusal* or *Failure Model* $\mathscr{F}$ [13, 11, 46] which makes slightly more identifications among processes than $\mathscr{R}$ – without affecting the expressibility of its liveness properties. In all these models a continuous hiding operator is available in full generality. The relationship between the models is established by (weak) homomorphisms.

The denotational models are related with a simple operational semantics based on transitions representing both external and internal process activity (Sects. 12 and 13). We show that the models $\mathscr{D}$, $\mathscr{R}$ and $\mathscr{F}$ are fully consistent with the operational semantics, whereas $\mathscr{C}$ and $\mathscr{T}$ are consistent only for divergence free processes. These results are obtained as an application of a general consistency theorem which relies only on the logical structure of observations.

Finally, we assess our approach and indicate further directions of research (Sect. 14).

2. Preliminaries

This section describes the general format of our programming language and denotational semantics (cf. [19, 20, 50]). It may be skipped now and consulted later when an unexplained basic notion is encountered.

Throughout this paper we consider a fixed set $(\xi\in)$ Idf of *identifiers* and a varying set $(f\in)\Sigma$ of *operator symbols* each one with a certain arity $n\geqq 0$. Σ is called a *signature*. Together with Idf it determines a simple programming

language, namely the set $(P, Q \in) \mathrm{Rec}(\Sigma)$ of *(recursive) terms* over Σ as defined by the following BNF-like syntax:

$$P ::= f(P_1, \ldots, P_n) \text{ where } f \text{ has arity } n \mid \xi \mid \mu\xi \cdot P.$$

The μ-construct $\mu\xi \cdot P$ denotes recursion [3]. It induces the usual notions of *free* and *bound* identifiers. For example, in the term $P = f(\xi, \mu\xi \cdot \xi)$ the first occurrence of ξ is free but the second and third one are bound. By $\mathrm{CRec}(\Sigma)$ we denote the set of all *closed* recursive terms, i.e. those without free identifiers. $\mathrm{FRec}(\Sigma)$ denotes the set of all *finite* terms, i.e. the closed terms $Q \in \mathrm{CRec}(\Sigma)$ without any occurrence of a μ-construct $\mu\xi \cdot P$ inside Q.

For terms $\mu\xi \cdot P$, $Q \in \mathrm{CRec}(\Sigma)$ let $P[Q/\xi]$ denote the result of *substituting* Q for every free occurrence of ξ in P. Since ξ is clear from the context $\mu\xi \cdot P$, we also write $P(Q)$ instead of $P[Q/\xi]$. This notation extends to n-fold substitution by defining $P^0(Q) = Q$ and $P^{n+1}(Q) = P(P^n(Q))$.

Let $\mathscr{D}$ be a set with a partial order $\sqsubseteq$. A subset $X \subseteq \mathscr{D}$ is *directed* if every finite subset of X has an upper bound in X. $\mathscr{D}$ is a *directed cpo* or just *cpo (complete partial order)* if it has a least element $\perp$ and if every directed subset $X \subseteq \mathscr{D}$ has a limit (least upper bound) $\bigsqcup X$ in $\mathscr{D}$. If $\mathscr{D}$ is a cpo, so is $\mathscr{D}^n = \mathscr{D} \times \ldots \times \mathscr{D}$ (n times), with component-wise ordering. The reason that we work here with directed sets rather than chains $X \subseteq \mathscr{D}$ is to ensure the general approximation result for denotational models stated in Proposition 2.1.

An operator $\phi: \mathscr{D} \to \xi$ from one cpo $\mathscr{D}$ into another cpo ξ is called *strict* if it preserves the least element, *monotonic* if it preserves the partial order $\sqsubseteq$, and *continuous* if it preserves limits of directed sets, i.e. if $\phi(\bigsqcup X) = \bigsqcup \phi(X)$ holds for every directed $X \subseteq \mathscr{D}$. Of course continuity implies monotonicity. We remark that an n-place operator $\phi: \mathscr{D}^n \to \mathscr{D}$ is continuous iff it is continuous in every place.

By Knaster-Tarski's fixed point theorem every monotonic operator $\phi: \mathscr{D} \to \mathscr{D}$ has a least fixed point $fix\,\phi$ in $\mathscr{D}$. If ϕ is also continuous, $fix\,\phi$ can be represented as

$$fix\,\phi = \bigsqcup \{\phi^n(\perp) \mid n \geqq 0\}$$

where $\phi^0(d) = d$ and $\phi^{n+1}(d) = \phi(\phi^n(d))$ holds for all $d \in \mathscr{D}$.

A *(denotational) model* $\mathscr{M}$ for $\mathrm{CRec}(\Sigma)$ consists of a cpo $\mathscr{D}_{\mathscr{M}}$ and a set $\{f_{\mathscr{M}} \mid f \in \Sigma\}$ of continuous operators $f_{\mathscr{M}}: \mathscr{D}_{\mathscr{M}} \times \ldots \times \mathscr{D}_{\mathscr{M}} \to \mathscr{D}_{\mathscr{M}}$. $\mathscr{M}$ induces a straightforward *denotational semantics* $\mathscr{M}[\![\cdot]\!]$ for $\mathrm{CRec}(\Sigma)$, in fact for $\mathrm{Rec}(\Sigma)$. Let $(\mathscr{V} \in) \mathrm{Val}$ be the set of *valuations*, i.e. mappings $\mathscr{V}: \mathrm{Idf} \to \mathscr{D}_{\mathscr{M}}$. Then

$$\mathscr{M}[\![\cdot]\!]: \mathrm{Rec}(\Sigma) \to (\mathrm{Val} \to \mathscr{D}_{\mathscr{M}})$$

is given by

(i) $\mathscr{M}[\![f(P_1, \ldots, P_n)]\!](\mathscr{V}) = f_{\mathscr{M}}(\mathscr{M}[\![P_1]\!](\mathscr{V}), \ldots, \mathscr{M}[\![P_n]\!](\mathscr{V}))$,

(ii) $\mathscr{M}[\![\xi]\!](\mathscr{V}) = \mathscr{V}(\xi)$,

(iii) $\mathscr{M}[\![\mu\xi \cdot P]\!](\mathscr{V}) = fix(\lambda d \cdot \mathscr{M}[\![P]\!](\mathscr{V}[d/\xi]))$

where $\mathscr{V}[d/\xi]$ is the valuation identical with $\mathscr{V}$ except at ξ where its value is $d \in \mathscr{D}_{\mathscr{M}}$. For closed terms $P \in \mathrm{CRec}(\Sigma)$ the value of $\mathscr{M}[\![P]\!](\mathscr{V})$ is independent of the particular valuation $\mathscr{V}$. Hence we write $\mathscr{M}[\![P]\!]$ instead of $\mathscr{M}[\![P]\!](\mathscr{V})$.

In the following we assume that there exists some 0-ary symbol $f \in \Sigma$ with $f_{\mathcal{M}} = \bot$. For simplicity let $\bot$ itself denote this symbol. For $P \in \mathrm{CRec}(\Sigma)$ let $P_\bot$ be that finite term which results from P by replacing *every* occurrence of a μ-construct $\mu\xi \cdot R$ in P by $\bot$. For $P, Q \in \mathrm{CRec}(\Sigma)$ we write $P \vdash Q$ if Q results from P by replacing *one* occurrence of a μ-construct $\mu\xi \cdot R$ by $R(\mu\xi \cdot R)$. Terms $Q_\bot$ with $P \vdash^* Q$ (reflexive, transitive closure) are called *finite (syntactic) approximations* of P. Note that $P \vdash^* Q$ implies

$$\mathcal{M}[\![P]\!] = \mathcal{M}[\![Q]\!] \quad \text{and} \quad \mathcal{M}[\![Q_\bot]\!] \sqsubseteq \mathcal{M}[\![P]\!].$$

Proposition 2.1. *In a denotational model $\mathcal{M}$ the equation*

$$\mathcal{M}[\![P]\!] = \bigsqcup \{\mathcal{M}[\![Q_\bot]\!] \mid P \vdash^* Q\}$$

holds for every term $P \in \mathrm{CRec}(\Sigma)$.

Proof. The operators in $\mathcal{M}$ are continuous and the family $\{\mathcal{M}[\![Q_\bot]\!] \mid P \vdash^* Q\}$ is directed. //

Thus every P is semantically the limit of its finite syntactic approximations. This representation is sometimes helpful when proving properties about denotational models $\mathcal{M}$.

An *(algebraic) law* in $\mathcal{M}$ is an equation

$$P = Q$$

with $P, Q \in \mathrm{CRec}(\Sigma)$ such that $\mathcal{M}[\![P]\!] = \mathcal{M}[\![Q]\!]$ holds. Let $\mathcal{M}$ be a model for $\mathrm{CRec}(\Sigma)$ and $\Sigma 1 \subseteq \Sigma$. Then the $\Sigma 1$-*reduct* $\mathcal{M} \restriction \Sigma 1$ is that model for $C\,\mathrm{Rec}(\Sigma 1)$ which consists of the cpo $\mathcal{D}_{\mathcal{M}}$ of $\mathcal{M}$ and the subset $\{f_{\mathcal{M}} \mid f \in \Sigma 1\}$ of operators.

Let $\mathcal{M}$, $\mathcal{N}$ be models for $\mathrm{CRec}(\Sigma)$. A *(weak) homomorphism* from $\mathcal{M}$ to $\mathcal{N}$ is an operator $\phi: \mathcal{D}_{\mathcal{M}} \to \mathcal{D}_{\mathcal{N}}$ such that

$$\phi(f_{\mathcal{M}}(d_1, \ldots, d_n)) = (\sqsubseteq) f_{\mathcal{N}}(\phi(d_1), \ldots, \phi(d_n))$$

holds for all $f \in \Sigma$ and $d_1, \ldots, d_n \in \mathcal{D}_{\mathcal{M}}$.

Proposition 2.2. *Let $\mathcal{M}$, $\mathcal{N}$ be as above and ϕ be a strict and continuous (weak) homomorphism from $\mathcal{M}$ to $\mathcal{N}$. Then*

$$\phi(\mathcal{M}[\![P]\!]) = (\sqsubseteq)\, \mathcal{N}[\![P]\!]$$

holds for every $P \in \mathrm{CRec}(\Sigma)$.

Proof. As above we may assume $\bot \in \Sigma$. Continuity of ϕ leads to

$$\phi(\mathcal{M}[\![P]\!]) = \bigsqcup \{\phi(\mathcal{M}[\![Q_\bot]\!]) \mid P \vdash^* Q\}$$

for every $P \in \mathrm{CRec}(\Sigma)$ due to Proposition 2.1. Strictness and the (weak) homomorphism property of ϕ yields

$$\phi(\mathcal{M}[\![Q_\bot]\!]) = (\sqsubseteq)\, \mathcal{N}[\![Q_\bot]\!]$$

for every Q with $P \vdash^* Q$ by structural induction (and the monotonicity of the operators in $\mathcal{N}$). //

Finally, we recall some set-theoretic notations. If A is a set, $\mathscr{P}(A)$ denotes the powerset $\mathscr{P}(A)=\{X|X\subseteq A\}$ of A and $|A|$ the cardinality of A. Besides the usual cartesian product $A\times B$ of sets A and B we consider the following *inner product* $\mathscr{A}\otimes\mathscr{B}$ for families $\mathscr{A}$ and $\mathscr{B}$ of sets:

$$\mathscr{A}\otimes\mathscr{B}=\{A\times B|A\in\mathscr{A}\text{ and }B\in\mathscr{B}\}.$$

Note that $\mathscr{A}\otimes\mathscr{B}\neq\mathscr{A}\times\mathscr{B}$. For a relation $g\subseteq A\times B$ and subsets $X\subseteq A$, $Y\subseteq B$ let $g(X)=\{b|\exists a\in X\colon agb\}$ and $g^{-1}(Y)=\{a|\exists b\in Y\colon agb\}$. For singleton sets we write $g(x)$ and $g^{-1}(y)$ instead of $g(\{x\})$ and $g^{-1}(\{y\})$. We call $g\subseteq A\times B$ *pre-image finite* if $g^{-1}(y)$ is finite for every $y\in B$, and *image finite* if $g(x)$ is finite for every $x\in A$. The product $g\circ h$ of relations $g\subseteq A\times B$ and $h\subseteq B\times C$ is defined by

$$a(g\circ h)c \text{ iff } \exists b\in B\colon agb\wedge bhc.$$

A relation $\rightarrow\subseteq A\times A$ is called *well-founded* if there is no infinite-to-the-left chain

$$\ldots\rightarrow a_3\rightarrow a_2\rightarrow a_1$$

of elements a_i in A. The reflexive, transitive closure of a relation $\rightarrow\subseteq A\times A$ is denoted by $\xrightarrow{*}$.

The notation $\mathscr{V}[d/\xi]$ used earlier is generalised to arbitrary mappings $f\colon A\rightarrow B$ and elements $a\in A$, $b\in B$ by defining $f[b/a]\colon A\rightarrow B$ as follows:

$$f[b/a](x)=\begin{cases}b & \text{if } x=a\\ f(x) & \text{otherwise.}\end{cases}$$

For sets X, Y, $Z\subseteq A$ we define the ternary *majority operator* $\cdot[\cdot]\cdot$ by

$$X[Y]Z=(X\cap Y)\cup(Y\cap Z)\cup(X\cap Z).$$

Thus an element is in $X[Y]Z$ iff it is in the majority (i.e. in at least two) of the sets X, Y, Z. This operator enjoys a number of algebraic properties. We state here only

$$\overline{X[Y]Z}=\bar{X}[\bar{Y}]\bar{Z}$$

where $\bar{}$ denotes the complement in A.

3. Communicating Processes

A process can engage in certain observable communications and internal actions. We are interested in networks of such processes which work in parallel and communicate with each other in a synchronised way. This section defines *Communicating Processes* as a language $\mathrm{CRec}(\Sigma)$ which describes how such networks can be constructed. The emphasis is here on analysing the fundamental concepts of communication and parallelism rather than on presenting a full programming notation as done for CSP [25] or OCCAM [30].

Formally, we start from a finite set or *alphabet* $(a,b\in)\,\mathrm{Comm}$ of *communications*. In OCCAM e.g. Comm is structured as $\mathrm{Comm}=\mathrm{Cha}\times\mathscr{M}$ where

Cha is a set of channel names and $\mathcal{M}$ is a set of messages. But for simplicity we shall not exploit such a structure here. The signature of Communicating Processes is given by the set

$$\Sigma = \{\mathsf{stop}, \mathsf{div}\} \cup \{a\rightarrow \mid a \in \mathrm{Comm}\} \cup \{\mathsf{or}, \Box\}$$
$$\cup \{\|_A \mid A \subseteq \mathrm{Comm}\} \cup \{\backslash b \mid b \in \mathrm{Comm}\}$$

of operator symbols. To fix the arities and some notational conventions we exhibit $\mathrm{Rec}(\Sigma)$:

$$P ::= \mathsf{stop} \mid \mathsf{div} \mid a\rightarrow P \mid P \text{ or } Q \mid P \Box Q \mid$$
$$P \|_A Q \mid P \backslash b \mid \xi \mid \mu\xi \cdot P.$$

The closed terms in $\mathrm{CRec}(\Sigma)$ are also called *processes.*

The intuitive meaning of processes is as follows: stop denotes a *deadlocked process* which neither engages in a communication nor in an internal action. div models the *diverging process* which pursues an infinite sequence of internal actions. $a\rightarrow P$ first communicates a and then behaves like P. This concept of *prefixing* is a restricted form of sequential composition. P or Q models *internal* or *local nondeterminism* [17]: it behaves like P or like Q, but the choice between them is nondeterministic and not controllable from outside. In contrast $P \Box Q$ models *external* or *global nondeterminism* [17]: the environment can control whether $P \Box Q$ behaves like P or like Q by choosing in its first step either to communicate with P or with Q. Compared with the original CSP [25] P or Q reflects the concept of a guarded command with true guards $[\mathsf{true}\rightarrow P \Box \mathsf{true}\rightarrow Q]$ and $P \Box Q$ corresponds to a guarded command with communication guards.

$P \|_A Q$ introduces *parallelism:* it behaves as if P and Q are working independently (asynchronously) except that all communications in the set A have to be synchronised. By varying its synchronisation set A parallel composition $\|_A$ reaches from arbitrary asynchrony ($\|_\phi$) to full synchrony ($\|_{\mathrm{Comm}}$). We remark that semantically asynchronous parallelism will be modelled by *interleaving.* While this simplifies the presentation of all our denotational models and the operational semantics, there is no inherent difficulty to consider also a semantics with explicit concurrency as shown in [36].

$P \backslash b$ behaves like P, but with all communications b *hidden* or *unobservable* from outside. Hiding brings the concept of *abstraction* into Communicating Processes. For simplicity we have restricted ourselves to a rather basic set of operators where for example explicit *renaming* $P[b/a]$ and full *sequential composition* are omitted. For a semantic treatment of these operators see e.g. [13].

Besides the full language $\mathrm{CRec}(\Sigma)$ we consider two sublanguages $C\mathrm{Rec}(\Sigma 1)$ and $\mathrm{CRec}(\Sigma 2)$ with $\Sigma 1 \subseteq \Sigma 2 \subseteq \Sigma$.

- $\Sigma 2$ is obtained from Σ by removing $\Box$.
- $\Sigma 1$ is obtained from $\Sigma 2$ by restricting parallel composition $\|_A$ to the case of $|A| \leqq 1$.

4. Observations and Specifications

It is quite easy to express the intuitive understanding of processes operationally in terms of transitions (see Sect. 12). But this formalisation has one severe disadvantage: it does not abstract from internal actions. Such an abstraction is essential for *hierarchical constructions* where one wants to compose large processes as networks of smaller ones and prove that after hiding certain connections these networks meet a given specification [37].

We develop therefore a different approach to the semantics of processes, called here specification-oriented semantics, which is based on the concepts of observation and specification. Our motivation is to express process correctness in the following uniform way: a process P satisfies a specification S, abbreviated as

$$P \text{ sat } S,$$

if every observation we can make of P is allowed by S. To realise this aim, we develop both a logical and an algebraic structure for observations. The logical structure tells us how we make an observation of a process and thus determines the notion of process correctness; and the algebraic structure provides a basis for denotational domains with continuous operators on sets of observations. These structures will be presented stepwise in several stages. Here we explain the simplest algebraic structure common to the refinements later on.

We are interested in observations we can make about the behaviour of a process P during its operation. This intuition leads us to postulate a relation $\longrightarrow\!\triangleright$ between observations which reflects their possible ordering in time: $x \longrightarrow\!\triangleright y$ means that observation y can be made immediately after x, without intervening observation.

Definition 4.1. A *simple observation space* is a structure $(\text{Obs}, \longrightarrow\!\triangleright)$ where $(x, y \in)\text{Obs}$ is a non-empty set of *observations* and $\longrightarrow\!\triangleright$ is a relation $\longrightarrow\!\triangleright \subseteq \text{Obs} \times \text{Obs}$ with:

(01) $\longrightarrow\!\triangleright$ is well-founded.

(02) $\longrightarrow\!\triangleright$ is pre-image finite.

(These notions are explained in Sect. 2.) //

Let

$$\text{Min} = \{x \in \text{Obs} \mid \neg\, \exists\, y \in \text{Obs}: y \longrightarrow\!\triangleright x\}.$$

Note that $\text{Min} \neq \emptyset$ because of condition (01) and $\text{Obs} \neq \emptyset$. By a *grounded chain* of length $n \geqq 0$ for x we mean a chain $x_0 \longrightarrow\!\triangleright \ldots \longrightarrow\!\triangleright x_n = x$ with $x_0 \in \text{Min}$. Due to (01) we can assign a *level* $\|x\|$ to every observation x:

$$\|x\| = \min\{n \geqq 0 \mid \exists \text{ grounded chain of length } n \text{ for } x\}.$$

Informally $\|x\|$ measures the observable progress a process has to make before the observation x is possible. Note that $x \longrightarrow\!\triangleright y$ implies $\|y\| \leqq \|x\| + 1$.

Example 4.2. Consider as observations the set

$$(s, t \in) \mathrm{Obs}_{\mathscr{T}} = \mathrm{Comm}^*$$

of *words* or *traces* over the finite communication alphabet Comm with ε denoting the empty trace. Define the relation $\rightarrowtriangle \subseteq \mathrm{Obs}_{\mathscr{T}} \times \mathrm{Obs}_{\mathscr{T}}$ as follows:

$$s \rightarrowtriangle t \text{ iff } \exists a \in \mathrm{Comm}: s \cdot a = t.$$

Then $(\mathrm{Obs}_{\mathscr{T}}, \rightarrowtriangle)$ is a simple observation space. And $s \overset{*}{\rightarrowtriangle} t$ holds iff s is a *prefix* of t, abbreviated $s \leqq t$. Note that here every trace s has exactly one grounded chain

$$\varepsilon \rightarrowtriangle \ldots \rightarrowtriangle s$$

and that the level $\|s\|$ coincides with the length of s. //

Given a simple observation space $(\mathrm{Obs}, \rightarrowtriangle)$ we can now talk about *(set-theoretic) specifications* over $(\mathrm{Obs}, \rightarrowtriangle)$: these are by definition simply sets $S \subseteq \mathrm{Obs}$ of observations. (A syntax for specifications will not be discussed in this paper.) We say S *allows* every observation $x \in S$. The idea is that S describes a set of nondeterministic possibilities of observations. This suggests the following *Smyth-like* ordering among specifications [48]:

$$S_1 \sqsubseteq S_2 \text{ iff } S_1 \supseteq S_2.$$

$S_1 \sqsubseteq S_2$ means that S_2 is *stronger* or *more deterministic* than S_1 or equivalently that S_1 is *weaker* than S_2. In particular Obs itself is the *weakest specification* allowing every observation.

We are aiming at denotational models $\mathscr{M}$ for $\mathrm{CRec}(\Sigma)$ which assign to every process $P \in \mathrm{CRec}(\Sigma)$ a specification $\mathscr{M}[\![P]\!]$ using ordering $\sqsubseteq$. But to do so the $\mathscr{M}[\![P]\!]$ must be special sets of observations, called here process specifications, which reflect the algebraic structure of Obs.

Definition 4.3. A *process specification* over $(\mathrm{Obs}, \rightarrowtriangle)$ is a subset $S \subseteq \mathrm{Obs}$ with:

(S1) S includes Min, i.e. $\mathrm{Min} \subseteq S$,

(S2) S is *generable*, i.e. $\forall x \in S - \mathrm{Min} \exists y \in S: y \rightarrowtriangle x$.

A *specification space* over $(\mathrm{Obs}, \rightarrowtriangle)$ is any set $(S, T \in) \mathrm{Spec} \subseteq \mathscr{P}(\mathrm{Obs})$ of process specifications which forms a cpo under the above ordering $\supseteq$. The set Spec consisting of *all* process specifications over $(\mathrm{Obs}, \rightarrowtriangle)$ is called the *full* specification space over $(\mathrm{Obs}, \rightarrowtriangle)$. (Since $\rightarrowtriangle$ is pre-image finite, it is indeed a cpo.) //

Example 4.4. Take $(\mathrm{Obs}_{\mathscr{T}}, \rightarrowtriangle)$ of Example 4.2. A subset $S \subseteq \mathrm{Comm}^*$ is called *prefix-closed* if $t \in S$ and $s \leqq t$ always imply $s \in S$. Then the set $\mathrm{Spec}_{\mathscr{T}}$ of all prefix-closed subsets $S \subseteq \mathrm{Comm}^*$ with $\varepsilon \in S$ is the full specification space over $(\mathrm{Obs}_{\mathscr{T}}, \rightarrowtriangle)$ [26]. //

If Spec_1 and Spec_2 are specification spaces over

$$(\mathrm{Obs}_1, \rightarrowtriangle_1) \quad \text{and} \quad (\mathrm{Obs}_2, \rightarrowtriangle_2),$$

hence cpo's, the cartesian product $\mathrm{Spec}_1 \times \mathrm{Spec}_2$ is of course a cpo with componentwise ordering $\supseteq$ (cf. Sect. 2) but not again a specification space. To achieve this, we consider the inner product $\mathrm{Spec}_1 \otimes \mathrm{Spec}_2$ (explained in Sect. 2). The sets $S \times T \in \mathrm{Spec}_1 \otimes \mathrm{Spec}_2$ are process specifications over the *product observation space*

$$(\mathrm{Obs}_1 \times \mathrm{Obs}_2, \rightarrowtriangle_{12})$$

where $\rightarrowtriangle_{12}$ is the following "interleaving relation" on $\mathrm{Obs}_1 \times \mathrm{Obs}_2$:

$$(x_1, x_2) \rightarrowtriangle_{12} (y_1, y_2) \quad \text{if either } x_1 \rightarrowtriangle_1 y_1 \text{ and } x_2 = y_2$$
$$\text{or } x_1 = y_1 \text{ and } x_2 \rightarrowtriangle_2 y_2$$

i.e. the pair makes a step when either component makes a step. Since the natural ordering $\supseteq$ on $\mathrm{Spec}_1 \otimes \mathrm{Spec}_2$ is isomorphic with the componentwise ordering on Spec_1 and Spec_2 in the sense that

$$S_1 \times S_2 \supseteq T_1 \times T_2 \text{ iff } S_1 \supseteq T_1 \text{ and } S_2 \supseteq T_2.$$

holds for all S_1, $T_1 \in \mathrm{Spec}_1$ and S_2, $T_2 \in \mathrm{Spec}_2$, it follows that the inner product $\mathrm{Spec}_1 \otimes \mathrm{Spec}_2$ is indeed a specification space over $(\mathrm{Obs}_1 \times \mathrm{Obs}_2, \rightarrowtriangle_{12})$.

5. Specification-oriented Semantics

We can now be precise about the desired form of semantics. Recall that Σ is the signature of Communicating Processes.

Definition 5.1. Let $\Sigma 0 \subseteq \Sigma$. A *specification-oriented model* for $\mathrm{CRec}(\Sigma 0)$ over $(\mathrm{Obs}, \rightarrowtriangle)$ consists of a specification space Spec over $(\mathrm{Obs}, \rightarrowtriangle)$ and a set $\{f_{\mathscr{M}} | f \in \Sigma 0\}$ of $\supseteq$-continuous operators on process specifications $S \in \mathrm{Spec}$. A *specification-oriented semantics* for $\mathrm{CRec}(\Sigma 0)$ is a denotational semantics $\mathscr{M}[\![\cdot]\!]$ induced by a specification-oriented model $\mathscr{M}$ for $\mathrm{CRec}(\Sigma 0)$. //

Correctness of processes $P \in \mathrm{CRec}(\Sigma 0)$ w.r.t. a specification $S \in \mathrm{Spec}$ is expressed by correctness "formulas" P sat S interpreted as follows:

$$\mathscr{M} \models P \text{ sat } S \text{ iff } \mathscr{M}[\![P]\!] \subseteq S.$$

By varying the structure of observations both safety and certain liveness properties of Communicating Processes can be expressed in this way. This clarifies the domains of our semantical models $\mathscr{M}$ and the notion of process correctness.

In the rest of this section we exploit the simple algebraic structure of observations and process specifications to derive some results for constructing $\supseteq$-continuous operators on specifications. This part may be skipped at a first reading and studied later when the Counter Model $\mathscr{C}$ is introduced.

Let Spec_1 and Spec_2 always denote specification spaces over $(\mathrm{Obs}_1, \rightarrowtriangle_1)$ and $(\mathrm{Obs}_2, \rightarrowtriangle_2)$. We wish to construct $\supseteq$-continuous operators

$$C_g \colon \mathrm{Spec}_1 \to \mathscr{P}(\mathrm{Obs}_2)$$

(working on process specifications) by starting from relations

$$g \subseteq \mathrm{Obs}_1 \times \mathrm{Obs}_2$$

that describe the desired effect of C_g "pointwise" for single observations.

The simplest way of achieving this is to take the pointwise extension $\mathcal{O}_g$: $\mathrm{Spec}_1 \to \mathcal{P}(\mathrm{Obs}_2)$ defined by

$$\mathcal{O}_g(S) = \{y \in \mathrm{Obs}_2 \mid \exists x \in S: x\, g\, y\} = g(S).$$

Clearly, $\mathcal{O}_g$ is $\supseteq$-monotonic, but not every relationship g induces a $\supseteq$-continuous $\mathcal{O}_g$.

Proposition 5.2. *If g is pre-image finite, the operator $\mathcal{O}_g$ is $\supseteq$-continuous.*

Proof. Since $\mathcal{O}_g$ is monotonic, it suffices to show that for every directed family $\{S_i \mid i \in I\}$ of $S_i \in \mathrm{Spec}_1$

$$\bigcap_i \mathcal{O}_g(S_i) \subseteq \mathcal{O}_g(\bigcap_i S_i)$$

holds. Consider some $y \notin \mathcal{O}_g(\bigcap_i S_i)$. Then $\bigcap_i S_i \cap g^{-1}(y) = \emptyset$. Since $g^{-1}(y)$ is finite and $\{S_i \mid i \in I\}$ is directed, there exists some $j \in I$ with $S_j \cap g^{-1}(y) = \emptyset$. Thus also $y \notin \bigcap_i \mathcal{O}_g(S_i)$. //

As we shall see in the following sections, most operators for Communicating Processes are induced by pre-image finite relations g. But the important hiding operators are not as illustrated by the next example.

Example 5.3. Take $\mathrm{Spec}_{\mathcal{T}}$ of Example 4.4. For observations (i.e. traces) s, $t \in \mathrm{Comm}^*$ we define

$$s\, g\, t \text{ iff } s \backslash b = t$$

where $s \backslash b$ is obtained from s by deleting or *hiding* all occurrences of b in s. Then g is not pre-image finite; and indeed $\mathcal{O}_g$: $\mathrm{Spec}_{\mathcal{T}} \to \mathcal{P}(\mathrm{Comm}^*)$ is not $\supseteq$-continuous as soon as $|\mathrm{Comm}| \geqq 2$ holds. Take e.g.

$$S_n = \{\varepsilon, b, \ldots, b^n\} \cup \{b^n s \mid s \in \mathrm{Comm}^*\}$$

for $n \geqq 0$. These S_n form a chain $S_0 \supseteq \ldots \supseteq S_n \supseteq \ldots$, but

$$\bigcap_n \mathcal{O}_g(S_n) = (\mathrm{Comm} - \{b\})^* \neq \{\varepsilon\} = \mathcal{O}_g(\bigcap_n S_n). \quad //$$

We present now an abstract analysis of such hiding operators based on the relation $\longrightarrow$ between observations. First we introduce a new operator $\mathcal{O}_g^\infty$: $\mathrm{Spec}_1 \to \mathcal{P}(\mathrm{Obs}_2)$ by

$$\mathcal{O}_g^\infty(S) = \{y' \mid \exists y \in \mathrm{Obs}_2\, \overset{\infty}{\exists} x \in S: (x\, g\, y \wedge y \xrightarrow{*}_2 y')\}$$

where $\overset{\infty}{\exists}$ means "there exist infinitely many". Informally speaking, $\mathcal{O}_g^\infty(S)$ diverges from y onwards if there are infinitely many $x \in S$ related to y by g. Instead

of $\mathcal{O}_g$ we will use the augmented operator C_g: $\mathrm{Spec}_1 \to \mathcal{P}(\mathrm{Obs}_2)$ defined by

$$C_g(S) = \mathcal{O}_g(S) \cup \mathcal{O}_g^\infty(S).$$

Continuity of C_g can be shown for certain well-behaved relations g.

Definition 5.4. A relation $g \subseteq \mathrm{Obs}_1 \times \mathrm{Obs}_2$ is called *level finite* if for every $y \in \mathrm{Obs}_2$ and $l \geqq 0$ there exist only finitely many $x \in g^{-1}(y)$ with level $\|x\| = l$. A relation $g \subseteq \mathrm{Obs}_1 \times \mathrm{Obs}_2$ is called *commutative* if $\xrightarrow{*}_1 \circ g \subseteq g \circ \xrightarrow{*}_2$ holds. //

Theorem 5.5. *If g is level finite and commutative, the operator $C_g = \mathcal{O}_g \cup \mathcal{O}_g^\infty$ is $\supseteq$-continuous.*

Proof. Since C_g is monotonic, it suffices again to show that for every directed family $\{S_i | i \in I\}$ of $S_i \in \mathrm{Spec}_1$

$$\bigcap_i C_g(S_i) \subseteq C_g(\bigcap_i S_i)$$

holds. Define $S = \bigcap_i S_i$ and consider some $y' \notin C_g(S)$. Let $Y = \{y | y \xrightarrow{*}_2 y'\}$. Then Y is finite because $\xrightarrow{*}_2$ is well-founded and pre-image finite. Thus by the definition of C_g:

(i) $S \cap g^{-1}(y') = \emptyset$.

(ii) $S \cap g^{-1}(Y)$ is finite.

Define $l = \max\{k | \exists x \in S \cap g^{-1}(Y): \|x\| = k\}$. Note that $l \in \mathbb{N}_0$ exists due to (ii). Since g is level finite, there are only finitely many $x \in g^{-1}(Y)$ with $\|x\| \leqq l+1$. Since $\{S_i | i \in I\}$ is directed, there exists some S_j such that for all x with $\|x\| \leqq l+1$:

(iii) $x \notin S_j \cap g^{-1}(y')$.

(iv) $x \in S_j \cap g^{-1}(Y)$ iff $x \in S \cap g^{-1}(Y)$.

Suppose now that $y' \in C_g(S_j)$ holds.

Case 1. $S_j \cap g^{-1}(y') \neq \emptyset$. By (iii) there is some $x \in S_j \cap g^{-1}(y')$ with $\|x\| > l+1$.

Case 2. $S_j \cap g^{-1}(y') = \emptyset$. Then $S_j \cap g^{-1}(Y)$ is infinite by the definition of C_g. Thus there is some $x \in S_j \cap g^{-1}(Y)$ with $x \notin S \cap g^{-1}(Y)$. By (iv) we conclude $\|x\| > l+1$.

Hence in both cases there is some $x \in S_j \cap g^{-1}(Y)$ with $\|x\| > l+1$. Consider some $y \in Y$ with $x g y$. Since S_j is generable, there is a grounded chain

$$x_0 \longrightarrow_1 \ldots \longrightarrow_1 x_m = x$$

in S_j. Then there exists some i with $0 \leqq i \leqq m$ and $\|x_i\| = l+1$. Clearly

$$x_i (\xrightarrow{*}_1 \circ g) y$$

holds. Commutativity of g implies

$$x_i (g \circ \xrightarrow{*}_2) y$$

and thus also $x_i \in S_j \cap g^{-1}(Y)$. By (iv) also $x_i \in S \cap g^{-1}(Y)$. But then $\|x_i\| \leqq l$ by the definition of l.

Contradiction

Hence $y' \notin \bigcap_i C_g(S_i)$, which is what had to be proved. //

Example 5.6. The hiding relation g of the previous Example 5.3 is level finite and commutative. Thus C_g is continuous: for the chain $S_0 \supseteq \ldots \supseteq S_n \supseteq \ldots$ we get

$$\bigcap_n C_g(S_n) = \text{Comm}^* = C_g(\bigcap_n S_n). \quad //$$

Above we considered only operators of one argument, but dealing with *several arguments* is easy: we just take the inner product $\otimes$ of the argument specification spaces according to Sect. 4. Note that neither Proposition 5.2 nor Theorem 5.5 claim that $\mathcal{O}_g$ or C_g yields a process specification in Spec_2 when applied to a process specification $S \in \text{Spec}_1$. This question will be treated separately in the individual cases.

6. The Counter Model $\mathscr{C}$

In the following sections we study a series of increasingly sophisticated specification-oriented models for Communicating Processes. These models vary in their choice of observations, and this will influence both the number of representable language operators and the induced notion of correctness.

Here we start with a very simple model for the sublanguage $\text{CRec}(\Sigma 1)$ of Communicating Processes where parallel composition is restricted to the case of $|A| \leqq 1$. We imagine that the only thing we can observe about a process P is how many times each communication $a \in \text{Comm}$ has occurred up to a given moment [28]. Formally, we define the set of *observations* by

$$\text{Obs}_{\mathscr{C}} = \{h \mid h: \text{Comm} \to \mathbb{N}_0\}$$

i.e. in every observation h, each communication a has a separate *counter* $h(a)$. $\text{Obs}_{\mathscr{C}}$ is a simple observation space with the following relation $\longrightarrow$:

$$h \longrightarrow h' \text{ iff } \exists a \in \text{Comm}: h' = h[h(a)+1/a]$$

i.e. h' differs from h in that exactly one counter is incremented by 1. (The variant notation $h[\ldots/a]$ is explained in Sect. 2.) Then $h \xrightarrow{*} h'$ means that $h(a) \leqq h'(a)$ holds for every $a \in \text{Comm}$ ($h \leqq h'$ for short). Let ZERO denote the constant mapping h with $h(a) = 0$ for every $a \in \text{Comm}$. As *process specifications* we take the full specification space $\text{Spec}_{\mathscr{C}}$ consisting of all generable subsets $S \subseteq \text{Obs}_{\mathscr{C}}$ with $\text{ZERO} \in S$.

The *Counter Model* $\mathscr{C}$ is now given by $\text{Spec}_{\mathscr{C}}$ and the following set $\{f_{\mathscr{C}} \mid f \in \Sigma 1\}$ of operators on process specifications S which formalise the intuitions about processes explained in Sect. 3 (since $\mathscr{C}$ remains constant throughout this section, we drop all indices $\mathscr{C}$ at operators $f_{\mathscr{C}}$, for instance we use stop instead of $\text{stop}_{\mathscr{C}}$):

(1) $$\mathsf{stop} = \{\mathrm{ZERO}\},$$

(2) $$\mathsf{div} = \mathrm{Obs}_{\mathscr{C}}.$$

In specification-oriented semantics we shall always identify diverging processes P with $S = \mathrm{Obs}$, the weakest specification in the $\supseteq$-ordering. Informally, a process P diverges if it engages in an infinite sequence of internal (i.e. hidden) actions. This operational concept of divergence will be defined later in Sect. 12 where an explicit symbol τ for hidden actions is used. An equivalent denotational definition of divergence will appear earlier in Sect. 8. We remark that divergence cannot only occur explicitly via the process div but also implicitly via recursion or hiding, for example in $\mu\xi \cdot \xi$ and $(\mu\xi \cdot b \to \xi)\backslash b$

(3) $$a \to S = \{\mathrm{ZERO}\} \cup \{h[h(a) + 1/a] \mid h \in S\}.$$

How do we ensure that this definition yields a well-defined and $\supseteq$-continuous operator? Using the notation of Sect. 5, we have

$$a \to S = \mathscr{O}_g(S) = \{h' \mid \exists h \in S : h g h'\}$$

where g is the following relation between observations:

$$h g h' \text{ iff } h = h' = \mathrm{ZERO} \text{ or } h' = h[h(a) + 1/a].$$

Since g is pre-image finite, the general Proposition 5.2 implies the $\supseteq$-continuity of $\mathscr{O}_g$. That $\mathscr{O}_g(S)$ is well-defined does not follow from such a general result but it is easy to see that with S also $\mathscr{O}_g(S)$ is generable.

(4) $$S_1 \text{ or } S_2 = S_1 \cup S_2.$$

This definition exhibits another typical point about specification-oriented semantics: due to our Smyth-like ordering $\sqsubseteq$ among specifications (internal) nondeterminism is modelled by set-theoretic union. Then

$$S_1 \sqsubseteq S_2 \text{ iff } S_1 \supseteq S_2 \text{ iff } S_1 = S_1 \text{ or } S_2$$

which accords with the idea that S_1 is more nondeterministic than S_2 (cf. Sect. 4).

(5) $$S_1 \|_A S_2 = \{h \mid \exists h_1 \in S_1\ \exists h_2 \in S_2 : \forall a \in A : h(a) = h_1(a) = h_2(a)$$

and

$$\forall a \notin A : h(a) = h_1(a) + h_2(a)\}.$$

This formalises the intuition that S_1 and S_2 work independently except for communications mentioned in the synchronisation set A. To check the continuity of $\|_A$, we consider the following relation g between the product $\mathrm{Obs}_{\mathscr{C}} \times \mathrm{Obs}_{\mathscr{C}}$ and $\mathrm{Obs}_{\mathscr{C}}$:

$$(h_1, h_2) g h \text{ iff } \forall a \in A : h(a) = h_1(a) = h_2(a)$$

and

$$\forall a \notin A : h(a) = h_1(a) + h_2(a).$$

Clearly g is pre-image finite and

$$S_1 \|_A S_2 = \mathcal{O}_g(S_1, S_2).$$

Thus Proposition 5.2 implies that $\|_A$ is indeed $\supseteq$-continuous.

What about well-definedness? The operator $\|_A$ preserves the generability of specifications S_1 and S_2 because of the restriction $|A| \leqq 1$ in $\Sigma 1$. For $|A| \geqq 2$ our simple definition of $\|_A$ does not necessarily ensure generability. For example, $S_1 = a \to b \to \mathsf{stop}$ and $S_2 = b \to a \to \mathsf{stop}$ denote generable specifications in $\mathrm{Spec}_{\mathscr{C}}$ but

$$S_1 \|_{\{a,b\}} S_2 = \{\mathrm{ZERO}\} \cup \{h \mid h(a) = h(b) = 1 \wedge \forall c \neq a, b: h(c) = 0\}$$

does not. Informally this is because we cannot observe the *relative timing* between different communications in the observations of the Counter Model $\mathscr{C}$. A similar problem, known as the *merge anomaly*, can arise in loosely coupled nondeterministic data flow network [10, 15].

(6) $S \backslash b$: we consider the relation $g \subseteq \mathrm{Obs}_{\mathscr{C}} \times \mathrm{Obs}_{\mathscr{C}}$ with:

$$(*) \qquad h g h' \text{ iff } h'(b) = 0 \text{ and } \forall a \neq b: h(a) = h'(a).$$

Intuitively, g hides all communications b in h. So can we take $S \backslash b = \mathcal{O}_g(S)$ as the hiding operator? No because as in Example 5.3, this definition is not $\supseteq$-continuous. In particular the relation g is not pre-image finite. But g is level finite and commutative. Thus Theorem 5.5 implies the $\supseteq$-continuity of the operator

$$C_g = \mathcal{O}_g \cup \mathcal{O}_g^{\infty}$$

which leads us to define $S \backslash b$ as follows:

$$S \backslash b = \{h \mid h(b) = 0 \wedge \exists n \geqq 0: h[n/b] \in S\} \cup \{h' \mid \exists h \leqq h' \overset{\infty}{\exists} n \geqq 0: h[n/b] \in S\}.$$

By the second clause, whenever for some observation h there are infinitely many communications b possible in S then $S \backslash b$ includes all observations h' with $h \leqq h'$. We remark that $S \backslash b$ preserves the generability of $S \in \mathrm{Spec}_{\mathscr{C}}$.

It is interesting to note that $S \backslash b = \mathcal{O}_g \cup \mathcal{O}_g^{\infty}$ is not $\supseteq$-continuous for arbitrary (non-generable) specifications $S \subseteq \mathrm{Obs}_{\mathscr{C}}$. Look e.g. at

$$S_n = \{\mathrm{ZERO}\} \cup \{h \mid h(b) \geqq n\}$$

for $n \geqq 0$. Of course

$$S_0 \supseteq \ldots \supseteq S_n \supseteq \ldots$$

holds with $\bigcap_n S_n = \{\mathrm{ZERO}\}$. But

$$\bigcap_n (S_n \backslash b) = \mathrm{Obs}_{\mathscr{C}} \neq \{\mathrm{ZERO}\} = (\bigcap_n S_n) \backslash b.$$

Thus *generability* of specifications is essential for the continuity of the above hiding operator $S \backslash b$.

This finishes the definition of the Counter Model $\mathscr{C}$. It induces a specification-oriented semantics $\mathscr{C}[\![\cdot]\!]$ for CRec($\Sigma 1$) as explained in Sects. 2 and 5. Recall that the semantics of recursive processes $\mu\xi \cdot P$ is defined by using least fixed points, i.e.

$$\mathscr{C}[\![\mu\xi \cdot P]\!] = \mathscr{C}[\![\mu\xi \cdot P]\!](\mathscr{V}) = fix(\lambda S \cdot \mathscr{C}[\![P]\!](\mathscr{V}[S/\xi])$$

for an arbitrary valuation $\mathscr{V}$. For example, since $\mathrm{Obs}_{\mathscr{C}}$ is the least fixed point of the identity function on $\mathrm{Spec}_{\mathscr{C}}$, we obtain

$$\mathscr{C}[\![\mu\xi \cdot \xi]\!] = \mathrm{Obs}_{\mathscr{C}}$$

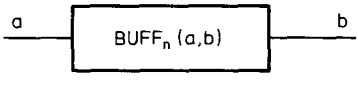
which agrees with our previous remarks on divergence.

Let us now study a more interesting application of the model $\mathscr{C}$: *a* chain of buffers.

Example 6.1. Consider for $n \geqq 1$ and $a, b \in \mathrm{Comm}$ the following specification:

$$\mathrm{BUFF}_n(a,b) = \{h \mid h(b) \leqq h(a) \leqq h(b) + n \\ \wedge \forall d \neq a, b : h(d) = 0\}.$$

Then $\mathrm{BUFF}_n(a,b)$ specifies a process which engages in communications a and b such that the number of b's never exceeds the number of a's and additionally the number of a's never exceeds the number of b's by more than n. This process can be visualised as an *n-place buffer*



which "inputs" a stream of a's and "outputs" a corresponding stream of b's in a buffered manner. Note that $\mathrm{BUFF}_n(a,b) \in \mathrm{Spec}_{\mathscr{C}}$.

It is easy to express 1-place buffers in CRec($\Sigma 1$): indeed with

$$P(a,b) = \mu\xi \cdot (a \to b \to \xi)$$

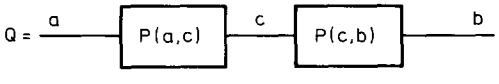
we get

$$\mathscr{C}[\![P(a,b)]\!] = \mathrm{BUFF}_1(a,b).$$

To construct larger n-place buffers hierarchically from simple 1-place buffers we use parallel composition and hiding. Let us demonstrate this for the case "$n = 2$". To build $\mathrm{BUFF}_2(a,b)$, we first construct a "chain" Q of two 1-place buffers:

$$Q = P(a,c) \|_{\{c\}} P(c,b)$$

or in graphical terms



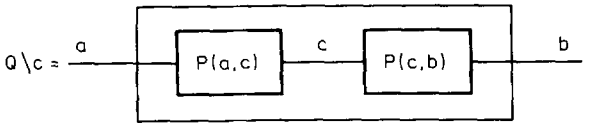
The resulting process behaves like a 2-place buffer except for the intermediate communications c:

$$\begin{aligned}\mathscr{C}[\![Q]\!] &= \mathscr{C}[\![P(a,c)]\!] \|_{\{c\}} \mathscr{C}[\![P(c,b)]\!] \\ &= \mathrm{BUFF}_1(a,c) \|_{\{c\}} \mathrm{BUFF}_1(c,b) \\ &= \{h \mid h(b) \leqq h(c) \leqq h(a) \leqq h(c)+1 \leqq h(b)+2 \\ &\quad \wedge \forall d \neq a,b,c : h(d)=0\}.\end{aligned}$$

To obtain the desired result we therefore internalise or hide all communications c:

$$Q \setminus c = (P(a,c) \|_{\{c\}} P(c,b)) \setminus c$$

or in graphical terms



This construction yields indeed

$$\mathscr{C}[\![(P(a,c) \|_{\{c\}} P(c,b)) \setminus c]\!] = \mathrm{BUFF}_2(a,b).$$

A "direct" construction of a 2-place buffer is given by

$$R = a \rightarrow \mu\xi \cdot ((a \rightarrow b \rightarrow \xi) \text{ or } (b \rightarrow a \rightarrow \xi))$$

with $\mathscr{C}[\![R]\!] = \mathrm{BUFF}_2(a,b)$. //

Discussion

In the example we dealt with semantic *equality*, e.g. we showed that $P(a,b)$ behaves exactly like a 1-place buffer, and $Q \setminus c$ exactly like a 2-place buffer. Let us now consider the notion of process correctness modelled by semantic *inclusion*, i.e.

$$\mathscr{C} \models P \text{ sat } S \text{ iff } \mathscr{C}[\![P]\!] \subseteq S.$$

Clearly

$$\mathscr{C} \models P(a,b) \text{ sat } \mathrm{BUFF}_1(a,b)$$

and

$$\mathscr{C} \models Q \setminus c \text{ sat } \mathrm{BUFF}_2(a,b).$$

But since $\mathrm{BUFF}_n(a,b) \subseteq \mathrm{BUFF}_{n+1}(a,b)$, we also have

$$\mathscr{C} \models P(a,b) \text{ sat } \mathrm{BUFF}_{n+1}(a,b)$$

and

$$\mathscr{C} \models Q \setminus c \text{ sat } \mathrm{BUFF}_{n+2}(a,b)$$

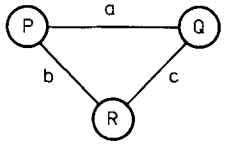
for all $n \geqq 0$. This means that we cannot use correctness reasoning based on $\mathscr{C} \models P(a,b)$ sat $\mathrm{BUFF}_n(a,b)$ to ensure that a buffer has a capacity of *at least n*.

What is worse, since

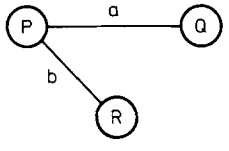
$$\mathscr{C} \models \text{stop sat } \mathrm{BUFF}_n(a,b),$$

we cannot even ensure that a buffer does anything at all. But the concept of "doing something" is already *a liveness property* of *a* process; it will be treated fully in Sects. 9 et seq.

Another shortcoming of the model $\mathscr{C}$ is due to the restriction $|A| \leqq 1$ in its signature $\Sigma 1$. If we picture processes $P_1, \ldots, P_n$ working in parallel as networks with $P_1, \ldots, P_n$ as nodes and synchronised communications between P_i and P_j as arcs, the restriction does not allow us to construct arbitrary cyclic networks. For example, we cannot construct the network



for it had to be defined as $(P \|_{\{a\}} Q) \|_{\{b,c\}} R$ or in a similar symmetric way. At best we can construct an *acyclic* or *tree-like* subnet of it, e.g. by $(P \|_{\{a\}} Q) \|_{\{b\}} R$ we get



with the arc c missing between Q and R.

7. The Trace Model $\mathscr{T}$

To treat CRec($\Sigma 2$) allowing *cyclic* networks of processes, we must be able to observe also the *relative order* of communications. This leads to the more informative observation set

$$(s, t \in) \, \mathrm{Obs}_{\mathscr{T}} = \mathrm{Comm}^*$$

of *traces* over Comm [26, 37]. $\mathrm{Obs}_{\mathscr{T}}$ induces the simple observation space $(\mathrm{Obs}_{\mathscr{T}}, \longrightarrow)$ and the full specification space $\mathrm{Spec}_{\mathscr{T}}$ explained in Examples 4.2 and 4.4.

The *Trace Model* $\mathscr{T}$ consists of $\mathrm{Spec}_{\mathscr{T}}$ and a set $\{f_{\mathscr{T}} \mid f \in \Sigma 2\}$ of operators defined as follows (again we drop indices $\mathscr{T}$ at $f_{\mathscr{T}}$):

(1) $\mathsf{stop} = \{\varepsilon\}$

(2) $\mathsf{div} = \mathrm{Obs}_{\mathscr{T}}$

(3) $a \rightarrow S = \{\varepsilon\} \cup \{as \mid s \in S\}$

(4) $S_1 \text{ or } S_2 = S_1 \cup S_2$

(5) $S_1 \|_A S_2 = \{s \mid \exists t_1 \in S_1, t_2 \in S_2 : s \in (t_1 \|_A t_2)\}$.

Here $t_1 \|_A t_2$ denotes the set of all successful interleavings of t_1 and t_2 with synchronising communications in A. Using the notation $a \cdot S = \{a \cdot s \mid s \in S\}$ we

can define $t_1 \|_A t_2$ inductively as follows:

(i) $\varepsilon \|_A \varepsilon = \{\varepsilon\}$,

(i) $as \|_A \varepsilon = \varepsilon \|_A as = \begin{cases} \emptyset & \text{if } a \in A \\ a \cdot (s \|_A \varepsilon) & \text{if } a \notin A, \end{cases}$

(iii) $as \|_A bt = bt \|_A as =$

$$\begin{cases} a \cdot (s \|_A t) & \text{if } a = b \in A \\ \emptyset & \text{if } a \neq b \wedge a, b \in A \\ a \cdot (s \|_A bt) & \text{if } a \notin A \wedge b \in A \\ a \cdot (s \|_A bt) \cup b \cdot (as \|_A t) & \text{if } a \notin A \wedge b \notin A \end{cases}$$

(6) $S \backslash b = \{s \backslash b \mid s \in S\}$

$$\cup \{(s \backslash b) t \mid t \in \mathrm{Comm}^* \wedge \forall n \geqq 0 : sb^n \in S\}$$

where $s \backslash b$ results from s by removing all occurrences of b in s. This completes the definition of the Trace Model $\mathscr{T}$. As with the previous model $\mathscr{C}$ all operator definitions of $\mathscr{T}$ can be derived systematically from appropriate relations g on traces and thus shown to be $\supseteq$-continuous (for $S \backslash b$ see also Example 5.3).

Relating $\mathscr{T}$ and $\mathscr{C}$

To relate the models $\mathscr{T}$ and $\mathscr{C}$ we consider the pointwise extension $\mathscr{O}_g$ of the following relation $g \subseteq \mathrm{Obs}_{\mathscr{T}} \times \mathrm{Obs}_{\mathscr{C}}$:

$$sgh \quad \text{iff } \forall a \in A : h(a) = a \# s$$

where $a \# s$ denotes the number of occurrences of a in s. Clearly $\mathscr{O}_g(S) \in \mathrm{Spec}_{\mathscr{C}}$ holds for every $S \in \mathrm{Spec}_{\mathscr{T}}$.

Proposition 7.1. *For every process $P \in \mathrm{CRec}(\Sigma 1)$ the equation $\mathscr{O}_g(\mathscr{T}[\![P]\!]) = \mathscr{C}[\![P]\!]$* holds.

Proof. By Proposition 2.2 it suffices to show that $\mathscr{O}_g$ is a strict and $\supseteq$-continuous homomorphism from the reduct $\mathscr{T} \restriction \Sigma 1$ to $\mathscr{C}$. Since g is pre-image finite, the $\supseteq$-continuity of $\mathscr{O}_g$ follows from Proposition 5.2. The homomorphism property of $\mathscr{O}_g$ (which implies here strictness) is easy to verify; only the parallel composition needs some care:

$$\mathscr{O}_g(S_1 \|_{A\mathscr{T}} S_2) = \mathscr{O}_g(S_1) \|_{A\mathscr{C}} \mathscr{O}_g(S_2)$$

depends on the restriction $|A| \leqq 1$ in $\Sigma 1$. Note: here we use the full notation $\|_{A\mathscr{T}}$ and $\|_{A\mathscr{C}}$ to distinguish between operators in $\mathscr{T}$ and $\mathscr{C}$. //

If we assume a channel structure $\mathrm{Comm} = Cha \times \mathscr{M}$ of communications, an interesting model $\mathscr{H}$ in between the Counter Model $\mathscr{C}$ and the Trace Model $\mathscr{T}$ is possible. The idea of $\mathscr{H}$ is that the relative order between communications can be observed if and only if they occur along the same channel. This leads to the following set of observations:

$$\mathrm{Obs}_{\mathscr{H}} = \{h \mid h\colon Cha \to \mathrm{Comm}^*\}$$

i.e. in every observation h, a separate history $h(ch) \in \mathrm{Comm}^*$ is kept for each channel $ch \in Cha$. We therefore talk of the *Channel History Model* $\mathscr{H}$. It is able to describe networks of processes *acyclically connected via channels*. Possible applications for $\mathscr{H}$ are buffers and protocols as demonstrated in [16].

8. The Divergence Model $\mathscr{D}$

In the Trace Model $\mathscr{T}$ of the previous section it can be proved that

$$\mathsf{div} \parallel_{\emptyset} P = \mathsf{div}$$

holds for every process P. This law accords perfectly with our intuition that an arbitrary interleaving of a process P with the diverging process div can itself pursue an infinite sequence of internal actions and thus be identified with div. On the other hand, we find that

$$\mathsf{div} \parallel_{\mathrm{Comm}} P = P$$

holds in $\mathscr{T}$, i.e. a full synchronisation of P with div ignores the possibility of divergence completely. This law seems unrealistic because the synchronisation set $A = \mathrm{Comm}$ should only restrict the observable behaviour of $\mathsf{div} \parallel_{\mathrm{Comm}} P$, not the internal actions (see also Sect. 13). Thus we would expect that on the contrary

$$(*) \qquad \mathsf{div} \parallel_{A} P = \mathsf{div}$$

holds for all synchronisation sets A and processes P. Similar problems arise in the simpler Counter Model $\mathscr{C}$.

What is the reason for this weakness of the models $\mathscr{C}$ and $\mathscr{T}$? In both models we identify diverging processes with the weakest specification $S = \mathrm{Obs}$. But in $\mathscr{C}$ and $\mathscr{T}$ this specification just models the concept of arbitrary observable nondeterminism. Thus we also identify diverging processes with non-diverging ones which exhibit arbitrary nondeterminism. For example, with $\mathrm{Comm} = \{a, b\}$

$$\mathsf{div} = \mu\xi \cdot ((a \to \xi) \text{ or } (b \to \xi))$$

holds in $\mathscr{T}$. This identification explains the unrealistic law $\mathsf{div} \parallel_{\mathrm{Comm}} P = P$ in $\mathscr{T}$.

In this section we improve the Trace Model $\mathscr{T}$ into a Divergence Model $\mathscr{D}$ where the law (*) is valid without qualification. The idea is to separate arbitrary nondeterminism from divergence. Thus in $\mathscr{D}$ only divergence will be modelled by the weakest specification $S = \mathrm{Obs}_{\mathscr{D}}$ (see also Definition 8.3).

To realise this idea, we introduce an explicit divergence symbol $\uparrow \notin \mathrm{Comm}$. If a process can diverge after a trace s of communications, we will include $s\uparrow$ in its set of observations. Of course, basic incomputability results tell us that we cannot expect to observe divergence effectively; the reason for introducing $\uparrow$ here is to obtain a semantic model where divergence is treated carefully.

Summarising, we take

$$\mathrm{Obs}_{\mathscr{D}} = \mathrm{Comm}^* \cup \{s\uparrow \mid s \in \mathrm{Comm}^*\}$$

as the set of observations with $\longrightarrow$ being the smallest relation satisfying

$$s \longrightarrow sa,\ s \longrightarrow s\uparrow$$

for all $s \in \mathrm{Comm}^*$ and $a \in \mathrm{Comm}$. Then $(\mathrm{Obs}_{\mathscr{D}}, \longrightarrow)$ is a simple observation space.

Next, we will use the divergence symbol $\uparrow$ to enforce the desired identification of divergence with $\mathrm{Obs}_{\mathscr{D}}$ in every process specification over $(\mathrm{Obs}_{\mathscr{D}}, \longrightarrow)$. More precisely, we wish to consider in our semantic model $\mathscr{D}$ only those sets $S \subseteq \mathrm{Obs}_{\mathscr{D}}$ as process specifications which also satisfy:

(**) $$s\uparrow \in S \text{ implies } sa, sa\uparrow \in S$$

for every $a \in \mathrm{Comm}$. The idea to require with (**) also a short of converse of the generability condition for process specifications is not only useful for the present model $\mathscr{D}$ but also fundamental in the following sections where we wish to deal with certain liveness properties. We therefore incorporate this idea into the general framework of observation and specification spaces, and call it the condition of extensibility.

Extensibility

The simplest definition of such a condition would be the literal converse of generability:

$$\forall x \in S - \mathrm{Max}\ \exists y \in S: x \longrightarrow y$$

where $\mathrm{Max} = \{x \mid \neg \exists y: x \longrightarrow y\}$. But this definition is too weak if we wish to express, as in (**), that *more than one* successor of x is to be present in S. We therefore extend the algebraic structure of observation spaces by a second relation $\longrightarrow\!\!\!\!\triangleright$ between single observations x and *sets* Y of observations.

Definition 8.1. An *observation space* is a structure $(\mathrm{Obs}, \longrightarrow, \longrightarrow\!\!\!\!\triangleright)$ where $(\mathrm{Obs}, \longrightarrow)$ is a simple observation space in the sense of Definition 4.1, and where $\longrightarrow\!\!\!\!\triangleright$ is a relation $\longrightarrow\!\!\!\!\triangleright \subseteq \mathrm{Obs} \times \mathscr{P}(\mathrm{Obs})$ such that

(03) $\longrightarrow\!\!\!\!\triangleright$ is image finite. //

As notation we introduce:

$$\mathrm{MAX} = \{x \in \mathrm{Obs} \mid \neg \exists Y \subseteq \mathrm{Obs}: x \longrightarrow\!\!\!\!\triangleright Y\}.$$

Simple observation spaces $(\mathrm{Obs}, \longrightarrow)$ will from now on be identified with observation spaces $(\mathrm{Obs}, \longrightarrow, \longrightarrow\!\!\!\!\triangleright)$ where the relation $\longrightarrow\!\!\!\!\triangleright$ is empty and thus $\mathrm{MAX} = \mathrm{Obs}$. If $\longrightarrow$ and $\longrightarrow\!\!\!\!\triangleright$ are understood, we also refer to Obs itself as the observation space. Next we adjust the notions of process specification and specification space.

Definition 8.2. A *process specification* over Obs is a set $S \subseteq \text{Obs}$ with:

(S1) S includes Min, i.e. $\text{Min} \subseteq S$,

(S2) S is *generable*, i.e. $\forall x \in S - \text{Min}\, \exists y \in S: y \longrightarrow\!\triangleright\, x$,

(S3) S is *extensible*, i.e. $\forall x \in S - \text{MAX}\, \exists Y \subseteq S: x \longrightarrow\!\!\triangleright\, Y$.

A *specification space* over Obs is any set $\text{Spec} \subseteq \mathscr{P}(\text{Obs})$ of process specifications which forms a cpo under $\supseteq$. Again the set Spec consisting of *all* process specifications over Obs is called the *full* specification space over Obs. (Since $\longrightarrow\!\triangleright$ is pre-image finite and $\longrightarrow\!\!\triangleright$ is image finite, it is indeed a cpo.) //

Note the symmetry between (S2) and (S3). If $\longrightarrow\!\!\triangleright$ is empty, Definition 8.2 reduces to Definition 4.3. In particular, every specification space over (Obs, $\longrightarrow\!\triangleright$, $\longrightarrow\!\!\triangleright$) is also a specification space over (Obs, $\longrightarrow\!\triangleright$). Thus our results about continuity in Sect. 5 remain valid because they rely only on the underlying structure (Obs, $\longrightarrow\!\triangleright$). Analogously to Definition 5.1 we define *specification-oriented models* $\mathscr{M}$ over (Obs, $\longrightarrow\!\triangleright$, $\longrightarrow\!\!\triangleright$).

Model $\mathscr{D}$ cont'd

Let us now continue with the Divergence Model $\mathscr{D}$. We take $\longrightarrow\!\!\triangleright$ to be the smallest relation between $\text{Obs}_{\mathscr{D}}$ and $\mathscr{P}(\text{Obs}_{\mathscr{D}})$ such that

$$s\uparrow \longrightarrow\!\!\triangleright \{sa, sa\uparrow \mid a \in \text{Comm}\}$$

holds for all $s \in \text{Comm}^*$. As process specifications we take the full specification space $\text{Spec}_{\mathscr{D}}$ over (Obs, $\longrightarrow\!\triangleright$, $\longrightarrow\!\!\triangleright$). Then every $S \in \text{Spec}_{\mathscr{D}}$ satisfies (**) as an instance of the general extensibility condition (S3) for S under $\longrightarrow\!\!\triangleright$. Note that "ordinary" traces $s \in S$ don't require any successors to be included in S. $\mathscr{D}$ is then determined by $\text{Spec}_{\mathscr{D}}$ and the following set $\{f_{\mathscr{D}} \mid f \in \Sigma 2\}$ of operators on process specifications S (we drop indices $\mathscr{D}$ and state only those definitions which *differ* from $\mathscr{T}$):

(2) $\text{div} = \text{Obs}_{\mathscr{D}}$

(5) $S_1 \|_A S_2 = \{s \mid \exists t_1 \in S_1, t_2 \in S_2: s \in t_1 \|_A t_2\}$
$\cup \{su \mid u \in \text{Obs}_{\mathscr{D}} \wedge \exists t_1 \in S_1, t_2 \in S_2:$
$(s \in t_1 \|_A t_2 \wedge (t_1\uparrow \in S_1 \vee t_2\uparrow \in S_2))\}$.

The second clause in the definition states that $S_1 \|_A S_2$ diverges as soon as either of its components diverge. Note that $S_1 \|_A S_2$ is a proper process specification and that the defining relation g with $S_1 \|_A S_2 = \mathscr{O}_g(S_1, S_2)$ is domain finite. This guarantees $\supseteq$-continuity by Proposition 5.2.

(6) $S \backslash b = \{s \backslash b \mid s \in S\}$
$\cup \{(s \backslash b)u \mid u \in \text{Obs}_{\mathscr{D}} \wedge \forall n \geqq 0: sb^n \in S\}$.

This is literally the definition of $S \backslash b$ from model $\mathscr{T}$ except that u ranges over $\text{Obs}_{\mathscr{D}}$ rather than $\text{Obs}_{\mathscr{T}} = \text{Comm}^*$. $S \backslash b$ is a proper process specification and can be proved $\supseteq$-continuous with help of Theorem 5.5.

$\mathscr{D}$ induces a specification-oriented semantics $\mathscr{D}[\![\cdot]\!]$ for $\mathrm{CRec}(\Sigma 2)$ in which the laws

(i) div or $P = P$ or div = div
(ii) div $\|_A P = P \|_A$ div = div
(iii) div$\backslash b$ = div

are true for all $P \in \mathrm{CRec}(\Sigma 2)$, $A \subseteq \mathrm{Comm}$ and $b \in \mathrm{Comm}$. (In the previous Trace Model $\mathscr{T}$ only (i) and (iii) hold.)

Relating $\mathscr{D}$ and $\mathscr{T}$

What is the relationship between the models $\mathscr{D}$ and $\mathscr{T}$? As explained earlier, the reason for achieving the law (**) in $\mathscr{D}$ is the careful distinction between arbitrary observable nondeterminism and divergence. This distinction can be made precise by considering the specification

$$S = \mathrm{Obs}_{\mathscr{T}} = \mathrm{Comm}^* \subseteq \mathrm{Obs}_{\mathscr{D}}$$

inside $\mathscr{D}$. S is a proper process specification over $\mathrm{Obs}_{\mathscr{D}}$; it is the weakest specification of a process without "diverging traces" $s\uparrow$. Thus whenever

$$\mathscr{D} \models P \text{ sat } \mathrm{Obs}_{\mathscr{T}}$$

holds, P is allowed to exhibit arbitrary observable nondeterminism but may not diverge. This motivates the following definition.

Definition 8.3. A process $P \in \mathrm{CRec}(\Sigma 2)$ is called *divergence free* if $\mathscr{D} \models P$ sat $\mathrm{Obs}_{\mathscr{T}}$ holds. //

Theorem 8.4. *For every process $P \in \mathrm{CRec}(\Sigma 2)$ the inclusion $\mathscr{T}[\![P]\!] \subseteq \mathscr{D}[\![P]\!]$ holds; for divergence free P the semantics $\mathscr{D}$ and $\mathscr{T}$ coincide: $\mathscr{D}[\![P]\!] = \mathscr{T}[\![P]\!]$.*

Proof. (i) $\mathscr{T}[\![P]\!] \subseteq \mathscr{D}[\![P]\!]$: The operator $\phi: \mathrm{Spec}_{\mathscr{D}} \to \mathrm{Spec}_{\mathscr{T}}$ with $\phi(S) = S \cap \mathrm{Comm}^*$ is a strict and continuous weak homomorphism (w.r.t. $\supseteq$) from $\mathscr{D}$ to $\mathscr{T}$. (Weakness is due to $\|_A$.) Thus Proposition 2.2 yields

$$\phi(\mathscr{D}[\![P]\!]) \supseteq \mathscr{T}[\![P]\!]$$

for every $P \in \mathrm{CRec}(\Sigma 2)$. By $\mathscr{D}[\![P]\!] \supseteq \phi(\mathscr{D}[\![P]\!])$, the claim follows.

(ii) $\mathscr{D}[\![P]\!] = \mathscr{T}[\![P]\!]$ for divergence free P:

We use finite syntactic approximations of P as defined in Sect. 2. Note that the general symbol $\perp$ of Sect. 2 is now div: thus we write P_{div} instead of $P_\perp$. First compare the operator definitions in $\mathscr{D}$ and $\mathscr{T}$ to realise that

$$(*) \qquad \mathscr{D}_+[\![f(P_1, \ldots, P_n)]\!] \subseteq f_{\mathscr{T}}(\mathscr{D}_+[\![P_1]\!], \ldots, \mathscr{D}_+[\![P_n]\!])$$

holds for all $f(P_1, \ldots, P_n) \in \mathrm{CRec}(\Sigma 2)$ where $\mathscr{D}_+[\![P]\!]$ denotes the divergence free part of a process P:

$$\mathscr{D}_+[\![P]\!] = \{t \in \mathscr{D}[\![P]\!] \mid \neg \exists s < t: s\uparrow \in \mathscr{D}[\![P]\!]\}.$$

Consider now a divergence free $P \in \mathrm{CRec}(\Sigma 2)$ and an arbitrary Q with $P \vdash\!\!\!\stackrel{*}{-} Q$. Note that Q can be written as

$$Q = Q^*[\mu\xi_1 \cdot R_1/\xi_1, \ldots, \mu\xi_n \cdot R_n/\xi_n]$$

where Q^* is a μ-free term with free identifiers $\xi_1, \ldots, \xi_n$ for which the recursive subterms $\mu\xi_1 \cdot R_1, \ldots, \mu\xi_n \cdot R_n$ of Q have been substituted. The following argument will use two valuations $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{\mathcal{T}}$ with

$$\mathcal{V}_{\mathcal{D}}(\xi_i) = \mathcal{D}_+[\![\mu\xi_i \cdot R_i]\!],$$
$$\mathcal{V}_{\mathcal{T}}(\xi_i) = \mathcal{T}[\![\mathsf{div}]\!]$$

for $i = 1, \ldots, n$. Since

$$(**) \qquad \mathcal{D}_+[\![\mu\xi_i \cdot R_i]\!] \subseteq \mathcal{T}[\![\mathsf{div}]\!]$$

holds, we get

$$\begin{aligned}\mathcal{D}_+[\![Q]\!] &= \mathcal{D}_+[\![Q^*[\mu\xi_1 \cdot R_1/\xi_1, \ldots, \mu\xi_n \cdot R_n/\xi_n]]\!] \\ &\subseteq \mathcal{T}[\![Q^*]\!](\mathcal{V}_{\mathcal{D}}) \text{ (by } (*)) \\ &\subseteq \mathcal{T}[\![Q^*]\!](\mathcal{V}_{\mathcal{T}}) \text{ (by } (**)) \\ &= \mathcal{T}[\![Q^*[\mathsf{div}/\xi_1, \ldots, \mathsf{div}/\xi_n]]\!] = \mathcal{T}[\![Q_{\mathsf{div}}]\!].\end{aligned}$$

Since Q was arbitrary with $P \vdash\!\!\!\stackrel{*}{-} Q$, we finally obtain

$$\begin{aligned}\mathcal{D}_+[\![P]\!] &= \bigcap\{\mathcal{D}_+[\![Q]\!] \mid P \vdash\!\!\!\stackrel{*}{-} Q\} \\ &\subseteq \bigcap\{\mathcal{T}[\![Q_{\mathsf{div}}]\!] \mid P \vdash\!\!\!\stackrel{*}{-} Q\} = \mathcal{T}[\![P]\!]\end{aligned}$$

due to Sect. 2. Thus, if P is divergence free

$$\mathcal{D}[\![P]\!] = \mathcal{D}_+[\![P]\!] \subseteq \mathcal{T}[\![P]\!]$$

holds, and (ii) follows with (i). //

9. Safety and Liveness Properties

What is the notion of process correctness induced by the previous Divergence Model? For processes $P \in \mathrm{CRec}(\Sigma 2)$ and specifications $S \in \mathrm{Spec}_{\mathcal{D}}$ we have

$$(*) \qquad \mathcal{D} \models P \text{ sat } S \text{ iff } \mathcal{D}[\![P]\!] \subseteq S.$$

Hence there is a particular process P which satisfies *every* specification S in $\mathcal{D}$, namely

$$P = \mathsf{stop}.$$

This indicates that $(*)$ expresses only *safety properties* [43] of P in the sense that P does nothing that is forbidden by S. For $(*)$ this means we can prove that

(a) P may only engage in the traces in S,
(b) P is divergence free, by choosing $S \subseteq \mathrm{Obs}_{\mathcal{T}}$.

(The simpler models $\mathscr{T}$ and $\mathscr{C}$ deal properly only with aspect (a) due to Theorem 8.4 and Proposition 7.1.) The situation has its analogue in the theory of *partial correctness* for sequential programs where the diverging program div plays the role of stop by satisfying every partial correctness formula $\{P\}\,\mathsf{div}\,\{Q\}$. In $\mathscr{D}$ the process div satisfies of course only the weakest specification $\mathrm{Obs}_{\mathscr{D}}$.

Let us now turn to the question of liveness properties. Intuitively, liveness means that a process is – independently of its internal activity – able to perform a certain predefined task [31]. In the following we propose an abstract framework for discussing this idea. Models incorporating liveness will follow in Sects. 10 and 11.

A *simple liveness property* is a pair

$$(T, L)$$

where T, L are sets of traces with $L \subseteq T$ and L prefix-closed. Informally a process

(**) $\qquad P$ satisfies (T, L)

iff the following holds:

(1) P is divergence free,

(2) P may only engage in the traces in T,

(3) P must (if the user insists on) engage in any trace in L – independently of any internal activity which might occur in between two successive communications in such a trace. (Thus L models the "task" mentioned above.)

A *safety property* is then a special case of (T, L) with $L = \emptyset$. To see the impact of the liveness condition (3), let us look at the processes

$$P1 = a \to \mathsf{stop} \,\Box\, c \to \mathsf{stop},$$
$$P2 = a \to \mathsf{div} \,\Box\, c \to \mathsf{stop},$$
$$P3 = (a \to \mathsf{stop} \,\Box\, b \to c \to \mathsf{stop}) \setminus b$$

and the simple liveness property (T, L) with

$$T = L = \{\varepsilon, a, c\}.$$

For which of the processes Pi do we have

$$Pi \text{ satisfies } (T, L)?$$

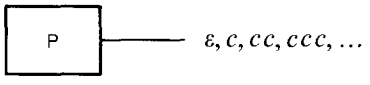
Clearly each Pi may only engage in the traces ε, a, c and thus satisfies condition (2). But only $P1$ and $P3$ are divergence free and hence satisfy condition (1). What about (3)? $P1$ indeed must engage in all the traces ε, a, c in L if the user insists on. But $P3$ cannot be forced to communicate a because it may engage in an internal action represented by the hidden communication b. After the hidden b the process $P3$ is only prepared for the communication c. Thus only $P1$ satisfies all the conditions of the simple liveness property (T, L).

In Sect. 10 we will exploit the extensibility condition of process specifications and translate every simple liveness property (T, L) into a process specification $S(T, L)$. Then (**) will be *defined* as

$$P \text{ sat } S(T, L)$$

in the sense of specification-oriented semantics. But at the moment the informal notion (**) should be sufficient for understanding the following examples.

Example 9.1. We wish to specify a process P which exactly sends an infinite stream of communications c:



We do this with the simple liveness property (T, L) where $T = L = \{c^n \mid n \geqq 0\}$. Then

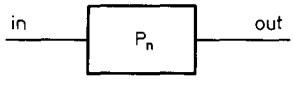
$$P \text{ satisfies } (T, L)$$

if firstly P does not engage in other communications than c due to T, and secondly P indeed engages in all traces

$$\varepsilon, c, cc, ccc, \ldots$$

due to L. //

Example 9.2. We are now able to reconsider Example 6.1 and express as a simple liveness property (T_n, L_n) the idea that a process P_n is a *buffer of capacity exactly n*. Instead of $a, b \in \text{Comm}$ we use here the suggestive names in, out$\in$Comm for the communications of P_n:



Take

$$T_n = L_n = \{t \in \{\text{in}, \text{out}\}^* \mid \forall s \leqq t: \text{out} \# s \leqq \text{in} \# s \leqq (\text{out} \# s) + n\}$$

where $\text{in} \# s$ ($\text{out} \# s$) denotes the number of in's (out's) in s (cf. Sect. 7). Then

$$P_n \text{ satisfies } (T_n, L_n)$$

if according to T_n the process P_n engages only in communications in and out such that the number of out's never exceeds the number of in's and the number of in's never exceeds the number of out's by more than n. This is the safety requirement for an n-place buffer known from Example 6.1.

But here we require more: P_n should also satisfy the liveness requirements described by L_n, viz.

(i) If the buffer P_n is "not full", i.e. if $\text{in} \# s < (\text{out} \# s) + n$, it should accept another in.

(ii) If the buffer P_n is "not empty", i.e. if $\text{out} \# s < \text{in} \# s$, it should be ready for another out.

Clearly, these requirements are not satisfied by the process stop any more (cf. Sect. 6). //

We generalise this concept of a simple liveness property as follows: a *(general) liveness property* is a non-empty set

$$\mathscr{L}$$

of simple liveness properties (T, L). We shall define in Sect. 10:

$$P \text{ satisfies } \mathscr{L} \text{ if } \forall \text{ deterministic behaviours } D \text{ of } P \\ \exists (T, L) \in \mathscr{L}\colon D \text{ satisfies } (T, L).$$

Simple liveness properties (T, L) can then be identified with singleton properties $\mathscr{L} = \{(T, L)\}$ because

$$P \text{ satisfies } (T, L) \text{ iff } P \text{ satisfies } \mathscr{L}.$$

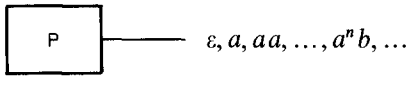
Intuitively, a simple liveness property fixes one particular task a process should perform whereas a general liveness property $\mathscr{L}$ describes only a general pattern of such a task. Then P satisfies $\mathscr{L}$ if - no matter how P behaves - it will always perform according to the required pattern.

Example 9.3. Surprisingly, we can view the concept of *deadlock freedom* (which is often classified as a safety property [43]) as a general liveness property

$$\mathscr{L} = \{(T, L) \mid T = \text{Comm}^* \text{ and } L \subseteq \text{Comm}^* \text{ prefix-closed with:} \\ \forall s \in L \exists t \in L\colon s < t\}.$$

In fact, P satisfies $\mathscr{L}$ iff after any amount of internal activity P is eventually prepared for some further communication which will extend its present communication trace s. (Since $T = \text{Comm}^*$ there is no restriction in which communications P may participate.) Thus deadlock freedom is viewed here as eventual progress which is of course a liveness property even in the informal sense of [43]. //

Example 9.4. We wish to specify a process P which can engage in communications a and b in arbitrary order, but which is certain to communicate b *eventually:*



We express this as a general liveness property

$$\mathscr{L} = \{(T, L_n) \mid n \geqq 0\}$$

with

$$T = \{a, b\}^* \quad \text{and} \quad L_n = \{\varepsilon, a, \ldots, a^n, a^n b\}.$$

Then P satisfies $\mathscr{L}$ if there is some $n \geqq 0$ such that P communicates b after n communications a; but it is not known in advance which n applies. Note that

after b has occurred, P may stop or engage in further communications a and b due to T. //

Example 9.5. Similarly, we can express the concept of a *bounded buffer* as a general liveness property

$$\mathscr{L} = \{(T_n, L_n) | n \geqq 0\}$$

with

$$T_n = L_n = \{t \in \{\text{in}, \text{out}\}^* | \forall s \leqq t: \text{out} \# s \leqq \text{in} \# s \leqq (\text{out} \# s) + n\}$$

as in Example 9.2. Now P satisfies $\mathscr{L}$ if there exists some "bound" n such that P behaves like an n-place buffer. Again it is not known which n applies. //

10. The Readiness Model $\mathscr{R}$

This section improves the Divergence Model $\mathscr{D}$ into a new model which can deal with simple (and a certain type of general) liveness properties: the *Readiness Model* $\mathscr{R}$. Moreover, $\mathscr{R}$ allows us to treat now the full language CRec(Σ) of Communicating Processes by distinguishing between external nondeterminism $\square$ and internal nondeterminism or. The idea of $\mathscr{R}$ is as follows: we assume that not only the "past" of a process can be observed via traces but also a limited part of the "future" via so-called *expectation* [18] or *ready sets* [22, 27] indicating which communications $b \in X$ can happen next. However, a ready set X can be observed only when the process has reached a "stable state" where all internal activity caused by hiding has ceased. This operational concept of stability will be defined later in Sect. 13 where an explicit symbol τ for hidden moves is used (see Definition 13.4).

Our observations have the form:

s	trace of successful communications,
sX	ready set X presented by the process after s,
$s\uparrow$	divergence after s.

Thus we obtain

$$\text{Obs}_{\mathscr{R}} = \{s, sX, s\uparrow | s \in \text{Comm}^* \wedge X \subseteq \text{Comm}\}.$$

Let s, t range over Comm* and X, Y over $\mathscr{P}$(Comm). The successor relation $\longrightarrow\!\!\triangleright$ is the smallest relation on $\text{Obs}_{\mathscr{R}}$ satisfying

$$s \longrightarrow\!\!\triangleright sX, \quad s \longrightarrow\!\!\triangleright sa, \quad s \longrightarrow\!\!\triangleright s\uparrow$$

$$sX \longrightarrow\!\!\triangleright sb \quad \text{for all } b \in X$$

for all $s \in \text{Comm}^*$, $X \subseteq \text{Comm}$ and $a \in \text{Comm}$. Relation $\longrightarrow\!\!\triangleright$ describes the behaviour of a process as follows: after a trace s the process can enter a stable state and display a ready set X, or it can (being in an unstable state) engage in some further communication a or it may diverge completely. Once in a stable state sX the process can engage only in the communications in the ready set X.

As extensibility relation $\twoheadrightarrow$ we take the smallest relation between $\mathrm{Obs}_{\mathscr{R}}$ and $\mathscr{P}(\mathrm{Obs}_{\mathscr{R}})$ such that

$$s \twoheadrightarrow \{sX\}$$
$$sX \twoheadrightarrow \{sb \mid b \in X\}$$
$$s\uparrow \twoheadrightarrow \{sX, sa, sa\uparrow \mid X \subseteq \mathrm{Comm} \wedge a \in \mathrm{Comm}\}$$

holds for all $s \in \mathrm{Comm}^*$ and $X \subseteq \mathrm{Comm}$. The process specifications of $\mathscr{R}$ are given by the full specification space $\mathrm{Spec}_{\mathscr{R}}$ over $(\mathrm{Obs}_{\mathscr{R}}, \rightarrow, \twoheadrightarrow)$. Then every $S \in \mathrm{Spec}_{\mathscr{R}}$ realises a *local liveness principle:* every $s \in S$ and $sX \in S$ with $X \neq \emptyset$ requires certain immediate successor observations to be present in S due to $\twoheadrightarrow$. Only observations

$$s\emptyset$$

have no successors and thus express *stoppage* or *deadlock*. The impact of this liveness principle will be studied later. First let us complete the definition of the Readiness Model $\mathscr{R}$ by the following set $\{f_{\mathscr{R}} \mid f \in \Sigma\}$ of operators (presented without index $\mathscr{R}$) on process specifications $S \in \mathrm{Spec}_{\mathscr{R}}$:

(1) $$\mathsf{stop} = \{\varepsilon, \varepsilon\emptyset\},$$

(2) $$\mathsf{div} = \mathrm{Obs}_{\mathscr{R}},$$

(3) $$a \rightarrow S = \{\varepsilon, \varepsilon\{a\}\} \cup \{as, as\triangle \mid s\triangle \in S\}.$$

Here and in the following clauses $\triangle$ always ranges over the set $\mathscr{P}(\mathrm{Comm}) \cup \{\uparrow\}$, i.e. $\triangle$ stands for either $\uparrow$ or some set $X \subseteq \mathrm{Comm}$.

(4) $$S_1 \quad \mathsf{or} \quad S_2 = S_1 \cup S_2,$$

(5) $$S_1 \Box S_2 = \{\varepsilon, \varepsilon(X \cup Y) \mid \varepsilon X \in S_1 \wedge \varepsilon Y \in S_2\} \cup \{\varepsilon\triangle \mid \varepsilon\uparrow \in S_1 \cup S_2\} \cup \{s, s\triangle \mid s \neq \varepsilon \wedge s\triangle \in S_1 \cup S_2\}.$$

The first clause states that $S_1 \Box S_2$ is initially ready for any communication in the union of the ready sets for its components S_1 and S_2. This enables us to model external nondeterminism. E.g.

$$(a \rightarrow P) \Box (b \rightarrow Q)$$

will have an initial ready set $\{a, b\}$ indicating that the environment can choose whether the process behaves like $a \rightarrow P$ or like $b \rightarrow Q$ by first communicating either a or b. In contrast

$$(a \rightarrow P) \quad \mathsf{or} \quad (b \rightarrow Q)$$

has two initial ready sets $\{a\}$ and $\{b\}$, and it depends on the process itself which one is presented to the environment.

This first clause in the definition of $S_1 \Box S_2$ requires the second clause to ensure that an initial divergence of S_1 or of S_2 causes an initial divergence of $S_1 \Box S_2$, represented by $\mathrm{Obs}_{\mathscr{R}}$ in the model.

(6)
$$\begin{aligned} S_1 \|_A S_2 = \{ & s, sX \mid \exists t_1 X_1 \in S_1, t_2 X_2 \in S_2 : \\ & s \in t_1 \|_A t_2 \wedge X = X_1 [\bar{A}] X_2 \} \\ \cup \{ & st, st\triangle \mid \exists t_1 \in S_1, t_2 \in S_2 : \\ & s \in t_1 \|_A t_2 \wedge (t_1\uparrow \in S_1 \vee t_2\uparrow \in S_2) \}. \end{aligned}$$

The first clause of the definition uses the majority operator of Sect. 2 for $\bar{A} = \mathrm{Comm} - A$:

$$X_1 [\bar{A}] X_2 = (X_1 \cap \bar{A}) \cup (X_2 \cap \bar{A}) \cup (X_1 \cap X_2)$$

formalises the idea that communications in A require the readiness of both S_1 and S_2 whereas for all other communications the readiness of S_1 or S_2 is sufficient.

(7) $S\backslash b$: we first introduce the hiding relation $g \subseteq \mathrm{Obs}_{\mathscr{R}} \times \mathrm{Obs}_{\mathscr{R}}$ which describes how observations about S are related to those about $S\backslash b$:

(i) $s\, g(s\backslash b)$

(ii) $sX\, g(s\backslash b) X$ provided $b \notin X$

(iii) $sX\, g(s\backslash b)$ provided $b \in X$

(iv) $s\uparrow g(s\backslash b)$

for all s and X. Clause (iii) may require a comment: since communication b has become internal in $S\backslash b$, the stable state sX of S with $b \in X$ has become unstable in $S\backslash b$ in the sense that b may occur autonomously, after which the process is no longer ready for any of the other communications of X. Therefore we cannot deduce any new ready set Y in this case and define

$$sX\, g(s\backslash b) \text{ provided } b \in X.$$

This definition agrees with the decisions taken in [18] and [27].

Since g is level finite and commutative, the operator $C_g = \mathscr{O}_g \cup \mathscr{O}_g^{\infty}$ is $\supseteq$-continuous by Theorem 5.5. This yields (after a slight simplification) as explicit definition:

$$\begin{aligned} S\backslash b = \{ & s\backslash b, (s\backslash b) X \mid sX \in S \wedge b \notin X \} \\ \cup \{ & (s\backslash b)t, (s\backslash b)t\triangle \mid \forall n \geqq 0 : sb^n \in S \} \end{aligned}$$

which is a proper process specification in $\mathrm{Spec}_{\mathscr{R}}$.

This completes the definition of the Readiness Model $\mathscr{R}$ which induces a specification-oriented semantics $\mathscr{R}[\![\cdot]\!]$ for the full language $\mathrm{CRec}(\Sigma)$ of Communicating Processes.

Expressing Liveness

Let us now investigate how this model can express liveness. For a simple liveness property (T, L) let $S_{\mathscr{R}}(T, L)$ be the following process specification in $\mathrm{Spec}_{\mathscr{R}}$:

$$\begin{aligned} S_{\mathscr{R}}(T, L) = T \cup \{ & sX \mid s \in T \wedge s \cdot X \subseteq T \\ & \wedge (\textbf{if } s \in L \textbf{ then } \mathrm{succ}_L(s) \subseteq X) \}. \end{aligned}$$

As abbreviations we use here $s \cdot X = \{sa | a \in X\}$ and $\mathrm{succ}_L(s) = \{a | sa \in L\}$ for $s \in \mathrm{Comm}^*$, $X \subseteq \mathrm{Comm}$ and prefix-closed $L \subseteq \mathrm{Comm}^*$.

We *define* now

$$P \text{ satisfies } (T, L) \text{ iff } \mathscr{R} \models P \text{ sat } S_{\mathscr{R}}(T, L)$$
$$\text{iff } \mathscr{R}[\![P]\!] \subseteq S_{\mathscr{R}}(T, L).$$

Let us check that this definition captures the conditions (1)-(3) of (T, L) as given in Sect. 9. Condition (1: P is divergence free) holds simply because the divergence symbol $\uparrow$ does not appear in $S_{\mathscr{R}}(T, L)$. Condition (2: P may only engage in the traces in T) also holds trivially because there are no other traces in $S_{\mathscr{R}}(T, L)$ than those of T.

Condition (3: P must engage in any trace in L - independently of any internal activity) holds by the extensibility condition imposed on the process specification $\mathscr{R}[\![P]\!]$. First recall that $\mathscr{R}[\![P]\!]$ includes the minimal observation ε. Now the extensibility relation $\twoheadrightarrow$ requires *some* successor observation

$$\varepsilon X$$

of ε to be present in $\mathscr{R}[\![P]\!]$. Because of $\mathscr{R}[\![P]\!] \subseteq S_{\mathscr{R}}(T, L)$ this observation εX must be allowed by $S_{\mathscr{R}}(T, L)$, i.e. satisfy the clause

$$\textbf{if } \varepsilon \in L \textbf{ then } \mathrm{succ}_L(\varepsilon) \subseteq X.$$

Thus no matter how the ready set X looks like, it must include all initial communications of the traces in L (provided $L \neq \emptyset$). Next the extensibility relation $\twoheadrightarrow$ requires *all* successor observations

$$a$$

of εX with $a \in X$ and hence all initial communications of L to be present in $\mathscr{R}[\![P]\!]$.

Since L is prefix-closed and the above arguments hold for any trace s in $\mathscr{R}[\![P]\!]$, not just ε, the inclusion $\mathscr{R}[\![P]\!] \subseteq S_{\mathscr{R}}(T, L)$ indeed ensures that, communication after communication, P must engage in any trace in L. Moreover, this "must" is independent of the internal activity of P since the ready pairs εX or more generally sX all refer to "stable states".

Returning to the processes $P1$-$P3$ and the simple liveness property (T, L) with

$$T = L = \{\varepsilon, a, c\}$$

of Sect. 9, we obtain

$$\mathscr{R}[\![P1]\!] = \{\varepsilon, a, c\} \cup \{\varepsilon\{a, c\}, a\emptyset, c\emptyset\},$$
$$\mathscr{R}[\![P2]\!] = \mathscr{R}[\![P1]\!] \cup a \cdot \mathrm{Obs}_{\mathscr{R}},$$
$$\mathscr{R}[\![P3]\!] = \{\varepsilon, a, c\} \cup \varepsilon\{c\}, a\emptyset, c\emptyset\}$$

and

$$S_{\mathscr{R}}(T, L) = \{\varepsilon, a, c\} \cup \{\varepsilon\{a, c\}, a\emptyset, c\emptyset\}.$$

Hence only for $i=1$ we have

$$\mathscr{R}[\![P\,i]\!] \subseteq S_{\mathscr{R}}(T, L).$$

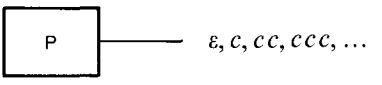
Thus in agreement with Sect. 9 only $P1$ satisfies (T, L). Next we consider more interesting liveness properties.

Example 10.1. To specify a process P which sends an infinite stream of communications c, we express the simple liveness property (T, L) of Example 9.1 in $\mathscr{R}$. This yields the following specification $\mathrm{SEND} = S_{\mathscr{R}}(T, L)$:

$$\mathrm{SEND} = \{c^n, c^n\{c\} \mid \text{where } n \geqq 0\}.$$

SEND says, no matter how many c's have already been sent, the process P should always be ready to send another c:

(*)



Then the extensibility condition of process specifications in $\mathrm{Spec}_{\mathscr{R}}$ forces every process P with

$$\mathscr{R} \models P \text{ sat } \mathrm{SEND}$$

to behave exactly like (*). A possible solution is

$$P = \mu\xi \cdot (c \to \xi).$$

In particular, the deadlocking process **stop** does not satisfy SEND. //

Example 10.2. To specify a buffer of capacity n we translate the simple liveness property (T_n, L_n) of Example 9.2 into the following specification $\mathrm{BUFF}_n = S_{\mathscr{R}}(T_n, L_n)$:

$$\begin{aligned}\mathrm{BUFF}_n = \{s, sX \mid s \in \{\mathrm{in}, \mathrm{out}\}^* \wedge X \subseteq \{\mathrm{in}, \mathrm{out}\} \\ \wedge \forall t \leqq s: \mathrm{out} \# t \leqq \mathrm{in} \# t \leqq (\mathrm{out} \# t) + n \\ \wedge (\textbf{if } \mathrm{in} \# s < (\mathrm{out} \# s) + n \textbf{ then } \mathrm{in} \in X \\ \wedge (\textbf{if } \mathrm{out} \# s < \mathrm{in} \# s \textbf{ then } \mathrm{out} \in X)\}.\end{aligned}$$

As in Example 6.1 we can construct buffers of capacity n hierarchically from buffers of capacity 1. Take

$$P_1(\mathrm{in}, \mathrm{out}) = \mu\xi \cdot (\mathrm{in} \to \mathrm{out} \to \xi)$$

and define inductively

$$P_{n+1}(\mathrm{in}, \mathrm{out}) = (P_1(\mathrm{in}, \mathrm{wire}) \|_{\{\mathrm{wire}\}} P_n(\mathrm{wire}, \mathrm{out})) \setminus \mathrm{wire}.$$

Then we have

$$\mathscr{R} \models P_n(\mathrm{in}, \mathrm{out}) \text{ sat } \mathrm{BUFF}_n.$$

Note that differently from Example 6.1

$$\mathrm{BUFF}_n \not\subseteq \mathrm{BUFF}_{n+1}.$$

Thus e.g. $P_1(\text{in}, \text{out})$ sat BUFF_2 is false in $\mathscr{R}$. Also note that direct constructions of buffers of capacity n involve external nondeterminism $\square$ rather than internal nondeterminism or: e.g. only with

$$R_2 = \text{in} \rightarrow \mu\xi \cdot (\text{in} \rightarrow \text{out} \rightarrow \xi \square \text{out} \rightarrow \text{in} \rightarrow \xi)$$

we get $\mathscr{R} \models R_2$ sat BUFF_2 (cf. Example 6.1). //

Next we investigate general liveness properties, given by non-empty sets $\mathscr{L}$ of simple liveness properties (T, L). As indicated in Sect. 9, we define

$$P \text{ satisfies } \mathscr{L} \text{ if } \forall \text{ deterministic behaviours } D \text{ of } P$$
$$\exists (T, L) \in \mathscr{L}: \; D \subseteq S_{\mathscr{R}}(T, L)$$

where a *deterministic behaviour* of P is a process specification $D \in \text{Spec}_{\mathscr{R}}$ with

$$D \subseteq \mathscr{R}[\![P]\!]$$

such that for every trace $s \in D$ there exists *exactly one* ready set X with $sX \in D$. Note that

$$\mathscr{R}[\![P]\!] = \bigcup \{D \mid D \text{ deterministic behaviour of } P\}.$$

Thus for liveness properties $\mathscr{L} = \{(T, L)\}$ we have

$$P \text{ satisfies } (T, L) \text{ iff } P \text{ satisfies } \mathscr{L}.$$

Therefore simple liveness properties (T, L) can be identified with singleton properties $\mathscr{L} = \{(T, L)\}$.

We introduce the following concept of expressiveness.

Definition 10.3. A liveness property $\mathscr{L}$ is expressible in a specification-oriented model $\mathscr{M}$ if there is a specification $S \in \text{Spec}_{\mathscr{M}}$ such that

$$P \text{ satisfies } \mathscr{L} \text{ iff } \mathscr{M} \models P \text{ sat } S$$

holds for every process $P \in \text{CRec}(\Sigma)$. We also say that S *expresses* $\mathscr{L}$ in $\mathscr{M}$. //

By definition, all simple liveness properties (T, L) are expressible in the Readiness Model $\mathscr{R}$. But what about general liveness properties $\mathscr{L}$?

Example 10.4. The introduction of observations $s\emptyset$ enable us to state and prove that a given process does *not* stop: consider the specification:

$$\text{LIVE} = \{s, sX \mid \text{where } X \neq \emptyset\}.$$

Then a process P with

$$\mathscr{R} \models P \text{ sat LIVE}$$

will after every trace s be ready to engage in some further communications, and thus never deadlock. Note that in fact LIVE expresses the general liveness property $\mathscr{L}$ of Example 9.3. //

The following proposition characterises the (limitations in) expressiveness of the Readiness Model $\mathscr{R}$.

Proposition 10.5. *$\mathscr{L}$ is expressible in $\mathscr{R}$ iff the following holds for all $\mathscr{P} \subseteq \mathrm{CRec}(\Sigma)$ and $Q \in \mathrm{CRec}(\Sigma)$: whenever*

$$\forall P \in \mathscr{P}\colon\ P \text{ satisfies } \mathscr{L}$$

and

$$\mathscr{R}[\![Q]\!] \subseteq \bigcup_{P \in \mathscr{P}} \mathscr{R}[\![P]\!]$$

then also Q satisfies $\mathscr{L}$.

Proof. **"only if":** by the definition of sat. **"if":** let $\mathscr{P} = \{P \mid P \text{ satisfies } \mathscr{L}\}$.

Case 1. $\mathscr{P} = \emptyset$. Then there is no $(T, L) \in \mathscr{L}$ and no process P such that P satisfies (T, L). Thus for arbitrary $(T, L) \in \mathscr{L}$ the specification $S = S_{\mathscr{R}}(T, L)$ expresses $\mathscr{L}$.

Case 2. $\mathscr{P} \neq \emptyset$. Then $S = \bigcup_{P \in \mathscr{P}} \mathscr{R}[\![P]\!]$ expresses $\mathscr{L}$. Note that indeed $S \in \mathrm{Spec}_{\mathscr{R}}$. //

Example 10.6. (i) The liveness property $\mathscr{L}$ of Example 9.4 modelling the concept of "eventually b" is not expressible in $\mathscr{R}$. Consider the processes

$$Q = \mu\xi \cdot (a \to \xi) \quad \text{and} \quad P_n = \underbrace{a \to \ldots \to a}_{n} \to b \to \mathsf{stop}$$

for $n \geqq 0$. Then P_n satisfies $\mathscr{L}$ and $\mathscr{R}[\![Q]\!] \subseteq \bigcup_n \mathscr{R}[\![P_n]\!]$, but Q does not satisfy $\mathscr{L}$. Thus $\mathscr{L}$ is not expressible in $\mathscr{R}$ by Proposition 10.5. This limitation in expressiveness is typical for any kind of *finitary* observation (see Sect. 13), not only for $\mathscr{R}$. Informally, we can say that the concept of eventuality is not finitely observable.

(ii) Similarly, we cannot express the concept of a bounded buffer modelled by the liveness property $\mathscr{L}$ of Example 9.5. Clearly

$$P_n(\mathrm{in}, \mathrm{out}) \text{ satisfies } \mathscr{L}$$

by the previous Example 10.2. Now consider

$$P_\infty = \mu\xi \cdot (\mathrm{in} \to (\xi \parallel_{\emptyset} \mathrm{out} \to \mathsf{stop})).$$

P_∞ expresses an *unbounded buffer:*

$$\begin{aligned} \mathscr{R}[\![P_\infty]\!] = \{s, sX \mid s \in \{\mathrm{in}, \mathrm{out}\}^* \wedge X \subseteq \{\mathrm{in}, \mathrm{out}\} \\ \wedge\, \forall t \leqq s\colon \mathrm{out} \# t \leqq \mathrm{in} \# t \\ \wedge\, \mathrm{in} \in X \\ \wedge\, (\textbf{if } \mathrm{out} \# s \leqq \mathrm{in} \# s \textbf{ then } \mathrm{out} \in X)\}. \end{aligned}$$

Thus $\mathscr{R}[\![P_\infty]\!] \subseteq \bigcup_n \mathscr{R}[\![P_n(\mathrm{in}, \mathrm{out})]\!]$ but P_∞ does not satisfy $\mathscr{L}$. Hence $\mathscr{L}$ is not expressible in $\mathscr{R}$. Again this limitation is true for any kind of finitary observation. Thus the concept of boundedness is not finitely observable.

(iii) Even much simpler liveness properties are not expressible in $\mathscr{R}$. Take e.g.

$$\mathscr{L} = \{(T, L_b), (T, L_c)\}$$

with

$$T=\text{Comm}^*, \quad L_b=\{\varepsilon, a, b, ab\} \quad \text{and} \quad L_c=\{\varepsilon, a, c, ac\}.$$

The idea of $\mathscr{L}$ is that b (or c) is possible after a only if it was also possible earlier as an alternative to a.

Consider now

$$P_1=(a\to b\to \text{stop})\,\square\, b\to \text{stop}$$

$$P_2=(a\to c\to \text{stop})\,\square\, c\to \text{stop}$$

$$Q=(a\to c\to \text{stop})\,\square\, b\to \text{stop}.$$

Then P_1 and P_2 satisfy $\mathscr{L}$ and $\mathscr{R}[\![Q]\!]\subseteq \mathscr{R}[\![P_1]\!]\cup \mathscr{R}[\![P_2]\!]$, but Q does not satisfy $\mathscr{L}$ as required by Proposition 10.5.

This limitation in expressiveness is typical for *trace-like* observations as used in $\mathscr{R}$. It could be overcome - if so desired - by using tree-like observations instead, but we decided here not to consider "what might have been" in our models for Communicating Processes. //

Relating $\mathscr{R}$ and $\mathscr{D}$

To relate the models $\mathscr{R}$ and $\mathscr{D}$ let $g\subseteq \text{Obs}_{\mathscr{R}}\times \text{Obs}_{\mathscr{D}}$ be the projection

$$sXgt \text{ iff } s=t$$

for observations sX and the identity otherwise. Then the pointwise extension $\mathscr{O}_g$ satisfies $\mathscr{O}_g(S)\in \text{Spec}_{\mathscr{D}}$ for every $S\in \text{Spec}_{\mathscr{R}}$.

Proposition 10.7. *For every process $P\in \text{CRec}(\Sigma 2)$ the equation $\mathscr{O}_g(\mathscr{R}[\![P]\!])=\mathscr{D}[\![P]\!]$ holds.*

Proof. By Propositions 2.2 and 5.2. //

Discussion

As indicated before, the idea of ready sets appeared earlier in the work of [18, 22, 27] but the details concerning hiding in our Readiness Model $\mathscr{R}$ are new. In particular, in [18] there is no abstraction from internal actions; they remain visible in the traces as little δ's.

It is interesting to note that the model $\mathscr{R}$ is well suited as a basis for implementing processes in a functional style [29, 45]. $\mathscr{R}$ has also been useful in establishing more advanced results about processes. For example, in [6] the concept of ready sets served as a stepping stone in relating a tree-like denotational semantics with a stream-like operational semantics. In [9] ready sets were helpful for proving a characterisation of the related failure semantics (see Sect. 11).

Models related to $\mathscr{R}$ are discussed in [47] and [40] (cf. Sect. 11). A restricted version of the Readiness Model forms also the basis in [38]. There

an extensibility condition of the form

$$\forall x \in S - \mathrm{Max}\, \exists y \in S\colon\ x \longrightarrow\!\!\triangleright\, y$$

appears whereas in $\mathscr{R}$ ready sets can require more than one successor of an observation to be present in a process specification S (cf. Sect. 8, Extensibility). Consequently [38] cannot deal with external nondeterminism.

11. The Failure Model $\mathscr{F}$

The Readiness Model $\mathscr{R}$ can express certain liveness properties by the use of ready sets displaying the next possible communications. This is done only in stable states where no hidden action is possible. Ready sets allow a careful distinction between internal and external nondeterminism. Moreover, ready sets even indicate a *third* kind of nondeterminism which can arise only by an application of the hiding operator on top of external nondeterminism.

An example is given by the process

$$P3 = (a \rightarrow \mathsf{stop} \,\Box\, b \rightarrow c \rightarrow \mathsf{stop}) \backslash b$$

considered already in Sects. 9 and 10. As semantics we obtain

$$\mathscr{R}[\![P3]\!] = \{\varepsilon, a, c\} \cup \{\varepsilon\{c\}, a\emptyset, c\emptyset\}$$

where only c but not the communication a appears in a ready set. This asymmetric kind of nondeterminism between a and c cannot be expressed as

$$Q = (a \rightarrow \mathsf{stop}) \quad \mathsf{or} \quad (c \rightarrow \mathsf{stop})$$

or

$$R = (a \rightarrow \mathsf{stop}) \,\Box\, (c \rightarrow \mathsf{stop})$$

because of

$$\mathscr{R}[\![Q]\!] = \{\varepsilon, a, c\} \cup \{\varepsilon\{a\}, \varepsilon\{c\}, a\emptyset, c\emptyset\}$$

and

$$\mathscr{R}[\![R]\!] = \{\varepsilon, a, c\} \cup \{\varepsilon\{a, c\}, a\emptyset, c\emptyset\}.$$

In fact, in the Readiness Model $\mathscr{R}$ the process $P3$ cannot be expressed without hiding (see Example 11.2).

The question arises whether this third, asymmetric kind of nondeterminism is worth having explicitly. Perhaps one could find a rationale by which $P3$ could safely be identified with a process involving only internal and external nondeterminism. What we do wish to maintain under such an identification are the liveness properties of $P3$.

More specifically the question is: can we find a model $\mathscr{M}$ for processes which expresses the same liveness properties as $\mathscr{R}$ but also admits reduction of the hiding operator in favour of internal and external nondeterminism. The first point is well understood from the previous section where the notion of expressibility of liveness properties $\mathscr{L}$ was defined for arbitrary models $\mathscr{M}$ (Definition 10.3) and characterised for the Readiness Model $\mathscr{R}$ (Proposi-

tion 10.5). But the second point requires a definition of what we mean by "reduction".

Recall that in any specification-oriented model $\mathcal{M}$ a process $P \in \mathrm{CRec}(\Sigma)$ can be semantically approximated as a limit of *finite*, i.e. non-recursive processes $Q_{\mathsf{div}} \in \mathrm{FRec}(\Sigma)$ via

$$\mathcal{M}[\![P]\!] = \bigcap \{\mathcal{M}[\![Q_{\mathsf{div}}]\!] \mid P \overset{*}{\mapsto} Q\}$$

(cf. Proposition 2.1). Therefore we shall restrict ourselves to finite processes when defining reduction. We call a finite process *primitive* if it is built from the signature

$$\Sigma p = \{\mathsf{stop}, \mathsf{div}\} \cup \{a \rightarrow \mid a \in \mathrm{Comm}\} \cup \{\mathsf{or}, \Box\}$$

not involving parallelism $\|_A$ and hiding $\backslash b$ of Σ. The set of all primitive finite processes is given by $\mathrm{FRec}(\Sigma p)$.

Definition 11.1. A model $\mathcal{M}$ for $\mathrm{CRec}(\Sigma)$ *admits reduction from* $\mathrm{FRec}(\Sigma)$ to $\mathrm{FRec}(\Sigma p)$ if for every finite $F \in \mathrm{FRec}(\Sigma)$ there exists some primitive finite $P \in \mathrm{Frec}(\Sigma p)$ such that the law

$$F = P$$

is true in $\mathcal{M}$. An operator $f \in \Sigma$ is called *reducible to* $\mathrm{FRec}(\Sigma p)$ *in* $\mathcal{M}$ if for all $P_1, \ldots, P_n \in \mathrm{FRec}(\Sigma p)$ there exists a $P \in \mathrm{FRec}(\Sigma p)$ such that

$$f(P_1, \ldots, P_n) = P$$

is true in $\mathcal{M}$. //

This explains our question above. Clearly $\mathcal{M}$ admits reduction from $\mathrm{Frec}(\Sigma)$ to $\mathrm{FRec}(\Sigma p)$ iff every operator $f \in \Sigma$ is reducible to $\mathrm{FRec}(\Sigma p)$ in $\mathcal{M}$. First let us understand precisely why the Readiness Model $\mathcal{R}$ does not admit reduction.

Example 11.2. Note that in $\mathcal{R}$ primitive finite processes $P \in \mathrm{FRec}(\Sigma p)$ satisfy the following property for all $s \in \mathrm{Comm}^*$ and $b \in \mathrm{Comm}$:

$$(1) \qquad sb \in \mathcal{R}[\![P]\!] \text{ implies } \exists X \subseteq \mathrm{Comm}: \; b \in X \wedge sX \in \mathcal{R}[\![P]\!]$$

i.e. before each communication b the process P reaches a stable state where a ready set X containing b is displayed.

Clearly our process $P3$ above does not satisfy (1) since the communication a does not appear in any ready set X. Hence $P3$ is not representable in $\mathrm{FRec}(\Sigma p)$.

What would be a good candidate in $\mathrm{FRec}(\Sigma p)$ to represent $P3$? We suggest

$$P = (a \rightarrow \mathsf{stop} \Box c \rightarrow \mathsf{stop}) \text{ or } c \rightarrow \mathsf{stop}$$

with

$$\mathcal{R}[\![P]\!] = \mathcal{R}[\![P3]\!] \cup \{\varepsilon\{a, c\}\}$$

because one can show that

$$P \text{ satisfies } \mathscr{L} \text{ iff } P3 \text{ satisfies } \mathscr{L}$$

for every liveness property $\mathscr{L}$. Thus identifying

(2) $$(a \to \mathsf{stop} \,\square\, b \to c \to \mathsf{stop}) \backslash b$$
$$= (a \to \mathsf{stop} \,\square\, c \to \mathsf{stop}) \text{ or } c \to \mathsf{stop}$$

does not affect the expressive power of our model in terms of liveness properties. //

The Model

We now explain a model which admits reduction to $\mathrm{FRec}(\Sigma p)$ essentially since the suggested law (2) is true: the *Refusal* or *Failure Model* $\mathscr{F}$ based on [13, 11, 46]. The initial idea of $\mathscr{F}$ looks quite different from $\mathscr{R}$. We imagine the following interactions to take place between a process P and its environment E: at any moment E can offer certain sets X of communications to P. The process has then three options to react to such an offer:

(*) either accept some communication $a \in X$
or refuse to accept any communication in X
or diverge completely.

Our observations record these interactions only until the first refusal of X has occurred:

sX trace of accepted communications together with a set X of communications which have been refused after s,
$s\uparrow$ possibility of divergence.

Thus we have

$$\mathrm{Obs}_{\mathscr{F}} = \{sX, s\uparrow \mid s \in \mathrm{Comm}^* \wedge X \subseteq \mathrm{Comm}\}.$$

Observations sX are called *failures* and the sets X *refusal sets* [13]. Failures of the form $s\emptyset$ are isomorphic to traces s considered explicitly in the Readiness Model $\mathscr{R}$; they represent interactions where no communication has been refused so far. As in Sect. 10 we let s, t range over Comm*, X, Y over $\mathscr{P}(\mathrm{Comm})$ and $\triangle$ over $\mathscr{P}(\mathrm{Comm}) \cup \{\uparrow\}$. As successor relation $\longrightarrow\!\triangleright$ we take the smallest relation on $\mathrm{Obs}_{\mathscr{F}}$ satisfying

$$s\emptyset \longrightarrow\!\triangleright sa\emptyset, \quad s\emptyset \longrightarrow\!\triangleright sX, \quad s\emptyset \longrightarrow\!\triangleright s\uparrow$$

for all $s \in \mathrm{Comm}^*$, $a \in \mathrm{Comm}$ and $X \subseteq \mathrm{Comm}$ with $X \neq \emptyset$.

The dynamic aspect of (*) is captured by the following extensibility relation $\longrightarrow\!\!\!\!\triangleright\!\triangleright$ between $\mathrm{Obs}_{\mathscr{F}}$ and $\mathscr{P}(\mathrm{Obs}_{\mathscr{F}})$:

$$s\emptyset \longrightarrow\!\!\!\!\triangleright\!\triangleright \mathrm{SUCC} \text{ iff } \forall X \subseteq \mathrm{Comm}: (\exists a \in X: sa\emptyset \in \mathrm{SUCC}$$
$$\vee \forall Y \subseteq X \text{ with } Y \neq \emptyset: sY \in \mathrm{SUCC})$$
$$s\uparrow \longrightarrow\!\!\!\!\triangleright\!\triangleright \{sX, sa\emptyset, sa\uparrow \mid X \subseteq \mathrm{Comm} \text{ with } X \neq \emptyset \wedge a \in \mathrm{Comm}\}.$$

Let us explain the more complex clause $s\emptyset$—⋈SUCC. Whenever $s\emptyset \in S$ holds for some process specification S, a whole set SUCC of successors of $s\emptyset$ must be present in S. This set SUCC looks as follows: for every given set $X \subseteq \mathrm{Comm}$ of communication either some $a \in X$ is accepted, i.e. $sa\emptyset \in S$ holds, or the whole set X together with every non-empty subset $Y \subseteq X$ is refused, i.e. $sX \in S$ and $sY \in S$ for $\emptyset \neq Y \subseteq X$ ($Y \neq \emptyset$ guarantees $s\emptyset \rightarrow sY$). Note that this definition reflects the informal description (*) above.

As process specifications of $\mathscr{F}$ we take the full specification space $\mathrm{Spec}_{\mathscr{F}}$ over $(\mathrm{Obs}_{\mathscr{F}}, \rightarrow, \text{—⋈})$. The Failure Model $\mathscr{F}$ consists of $\mathrm{Spec}_{\mathscr{F}}$ and the following set $\{f_{\mathscr{F}} \mid f \in \Sigma\}$ of operators (again we drop the index $\mathscr{F}$):

(1) $$\mathsf{stop} = \{\varepsilon X \mid X \subseteq \mathrm{Comm}\}.$$

The deadlocking process can refuse any set X.

(2) $$\mathsf{div} = \mathrm{Obs}_{\mathscr{F}},$$

(3) $$a \rightarrow S = \{\varepsilon X \mid a \notin X\} \cup \{as\,\triangle \mid s\,\triangle \in S\}.$$

In its first step $a \rightarrow S$ can refuse any communication except a.

(4) $$S_1 \text{ or } S_2 = S_1 \cup S_2$$

(5) $$\begin{aligned} S_1 \,\Box\, S_2 = \; & \{\varepsilon X \mid \varepsilon X \in S_1 \cap S_2\} \\ & \cup \{\varepsilon\,\triangle \mid \varepsilon\uparrow \in S_1 \cup S_2\} \\ & \cup \{s\,\triangle \mid s \neq \varepsilon \wedge s\,\triangle \in S_1 \cup S_2\}. \end{aligned}$$

In its first step $S_1 \,\Box\, S_2$ can refuse only communications that both S_1 and S_2 can refuse. Afterwards $S_1 \,\Box\, S_2$ behaves like S_1 or like S_2 depending on whether the first accepted communication belongs to S_1 or S_2.

(6) $$\begin{aligned} S_1 \|_A S_2 = \; & \{sX \mid \exists t_1 X_1 \in S_1,\, t_2 X_2 \in S_2: \\ & \qquad s \in t_1 \|_A t_2 \wedge X = X_1[A]X_2\} \\ & \cup \{st\,\triangle \mid \exists t_1 \emptyset \in S_1,\, t_2 \emptyset \in S_2: \\ & \qquad s \in t_1 \|_A t_2 \wedge (t_1\uparrow \in S_1 \vee t_2\uparrow \in S_2)\} \end{aligned}$$

where the majority operator

$$X_1[A]X_2 = (X_1 \cap A) \cup (X_2 \cap A) \cup (X_1 \cap X_2)$$

represents the idea that refusal of communications outside A requires refusal of both S_1 and S_2 whereas communications inside A can already be refused if S_1 or S_2 refuse them.

(7) $$\begin{aligned} S \backslash b = \; & \{(s \backslash b)X \mid s(X \cup \{b\}) \in S\} \\ & \cup \{(s \backslash b)t\,\triangle \mid \forall n \geqq 0: sb^n\emptyset \in S\}. \end{aligned}$$

Note that $S \backslash b$ can refuse a set X only if $S \backslash b$ has become stable, i.e. internal communications b of S are also refused.

As with the previous models these operator definitions yield proper process specifications, and they can be shown $\supseteq$-continuous by the methods of Sect. 5. This finishes the definition of $\mathscr{F}$.

Remark 11.3. A set $S \subseteq \mathrm{Obs}_{\mathscr{F}}$ is a process specification in $\mathrm{Spec}_{\mathscr{F}}$ iff the following holds:

(i) $\varepsilon\emptyset \in S$,

(ii) $stX \in S$ implies $s\emptyset \in S$,

(iii) $s\emptyset \in S \wedge \forall a \in X: sa\emptyset \notin S$ implies $sX \in S$,

(Iv) $s\uparrow \in S$ implies $st\triangle \in S$ for all $t, \triangle$.

If moreover $S = \mathscr{F}[\![P]\!]$ holds for some $P \in \mathrm{CRec}(\Sigma)$, S satisfies additionally:

(v) $sX \in S \wedge Y \subseteq X$ implies $sY \in S$,

(vi) $sX \in S \wedge sa\emptyset \notin S$ implies $s(X \cup \{a\}) \in S$. //

Relating $\mathscr{F}$ and $\mathscr{R}$

Let us first establish the relationship between the new model $\mathscr{F}$ and the previous Readiness Model $\mathscr{R}$. This is done very simply by the following relation $g \subseteq \mathrm{Obs}_{\mathscr{R}} \times \mathrm{Obs}_{\mathscr{F}}$ with

$$\begin{aligned} sX\, g\, sZ \quad &\text{iff } Z \subseteq \bar{X} \\ s\uparrow\, g\, s\uparrow \end{aligned}$$

which interprets refusal sets $\mathbf{Z}$ as the *downward closures* of the complements $\bar{X} = \mathrm{Comm} - X$ of ready sets X. By Remark 11.3 the pointwise extension $\mathscr{O}_g$ maps every $S \in \mathrm{Spec}_{\mathscr{R}}$ into a process specification $\mathscr{O}_g(S) \in \mathrm{Spec}_{\mathscr{F}}$.

Proposition 11.4. *For every process $P \in C\,\mathrm{Rec}(\Sigma)$ the equation $\mathscr{O}_g(\mathscr{R}[\![P]\!]) = \mathscr{F}[\![P]\!]$* holds.

Proof. To apply Proposition 2.2 we have to show that $\mathscr{O}_g$ is a (strict and) $\supseteq$-continuous homomorphism from $\mathscr{R}$ to $\mathscr{F}$. Pre-image finiteness of g implies the $\supseteq$-continuity of $\mathscr{O}_g$ by Proposition 5.2. Checking the homomorphism property of $\mathscr{O}_g$ boils down to a simple calculation with downward closures of complements. For example the crucial argument to show

$$\mathscr{O}_g(S_1 \|_{A_{\mathscr{R}}} S_2) = \mathscr{O}_g(S_1) \|_{A_{\mathscr{F}}} \mathscr{O}_g(S_2)$$

for all $S_1, S_2 \in \mathrm{Spec}_{\mathscr{R}}$ is as follows:

$$Z \subseteq \overline{X_1[\bar{A}]X_2} \quad \text{iff } \exists Z_1 \subseteq \bar{X}_1\ \exists Z_2 \subseteq \bar{X}_2\colon\ Z = Z_1[A]Z_2. \quad //$$

Clearly there is also a direct homomorphism ϕ from the reduct $\mathscr{F}\uparrow\Sigma 2$ to the Divergence Model $\mathscr{D}$ analogously to Proposition 10.7. By Proposition 11.4 every law $P = Q$ of $\mathscr{R}$ holds also in $\mathscr{F}$. But what are the additional identifications induced in $\mathscr{R}$ by the homomorphism $\mathscr{O}_g$?

Definition 11.5. A process specification $S \in \mathrm{Spec}_{\mathscr{R}}$ in the Readiness Model $\mathscr{R}$ is called *convex closed* [12, 40] if

$$sX \in S \quad \text{and} \quad X \subseteq Y \subseteq \{a \mid sa \in S\}$$

always imply

$$sY \in S.$$

For $S \in \mathrm{Spec}_{\mathscr{R}}$ let $\mathrm{con}(S)$ denote the w.r.t. $\supseteq$ strongest convex closed specification with $\mathrm{con}(S) \supseteq S$. //

Note that $\mathrm{con}(S) \in \mathrm{Spec}_{\mathscr{R}}$. Of course S is convex closed iff $\mathrm{con}(S) = S$. Returning to Example 11.2 with

$$P3 = (a \to \mathsf{stop} \,\Box\, b \to c \to \mathsf{stop}) \setminus b$$

and

$$P = (a \to \mathsf{stop} \,\Box\, c \to \mathsf{stop}) \text{ or } c \to \mathsf{stop},$$

it is easy to check that $\mathscr{R}[\![P]\!]$ is convex closed but $\mathscr{R}[\![P3]\!]$ is not. Indeed

$$\mathscr{R}[\![P]\!] = \mathrm{con}(\mathscr{R}[\![P3]\!]).$$

Lemma 11.6. *For $S_1, S_2 \in \mathrm{Spec}_{\mathscr{R}}$ the following holds:*

$$\mathscr{O}_g(S_1) \subseteq \mathscr{O}_g(S_2) \quad \textit{iff} \quad \mathrm{con}(S_1) \subseteq \mathrm{con}(S_2).$$

Proof. **"if"**: Consider $sZ \in \mathscr{O}_g(S_1)$. Then $sX_1 \in S_1$ with $Z \subseteq \bar{X}_1$ for some $X_1 \subseteq \mathrm{Comm}$. Note that $sX_1 \in \mathrm{con}(S_1) \subseteq \mathrm{con}(S_2)$. Hence there exists some $X_2 \subseteq X_1$ with $sX_2 \in S_2$. Since $Z \subseteq \bar{X}_1 \subseteq \bar{X}_2$, also $sZ \in \mathscr{O}_g(S_2)$.

"only if": Let $sY \in \mathrm{con}(S_1)$. Then $sX_1 \in S_1$ and $X_1 \subseteq Y \subseteq \{a \mid sa \in S_1\}$ for some $X_1 \subseteq \mathrm{Comm}$. Then $sZ \subseteq \mathscr{O}_g(S_1) \subseteq \mathscr{O}_g(S_2)$ for all $Z \subseteq \bar{X}_1$. Thus there exists some $X_2 \subseteq X_1 \subseteq Y$ with $sX_2 \in S_2$. Since $\mathscr{O}_g(S_1) \subseteq \mathscr{O}_g(S_2)$ implies $\{a \mid sa \in S_1\} \subseteq \{b \mid sb \in S_2\}$, we have $X_2 \subseteq Y \subseteq \{b \mid sb \in S_2\}$ and therefore $sY \in \mathrm{con}(S_2)$. //

Combining Proposition 11.4 and Lemma 11.6 yields:

Corollary 11.7. *For processes $P, Q \in C\,\mathrm{Rec}(\Sigma)$ we have:*

$$\mathscr{F}[\![P]\!] = \mathscr{F}[\![Q]\!] \quad \textit{iff} \quad \mathrm{con}(\mathscr{R}[\![P]\!]) = \mathrm{con}(\mathscr{R}[\![Q]\!]).$$

Expressing Liveness

Thus the Failure Model $\mathscr{F}$ can be considered as identifying every process specifications S of the Readiness Model $\mathscr{R}$ with its convex closure $\mathrm{con}(S)$. This characterisation of $\mathscr{F}$ in terms of $\mathscr{R}$ allows us now to show that all liveness properties expressible in $\mathscr{R}$ can also be expressed in $\mathscr{F}$. First we state:

Proposition 11.8. *In general $\mathscr{R}$ is more expressive than $\mathscr{F}$ in the following sense:*

(i) *For every $S_{\mathscr{F}} \in \mathrm{Spec}_{\mathscr{F}}$ there exists some $S_{\mathscr{R}} \in \mathrm{Spec}_{\mathscr{R}}$ with*

(*) $\mathscr{R} \models P \text{ sat } S_{\mathscr{R}}$ *iff* $\mathscr{F} \models P \text{ sat } S_{\mathscr{F}}$

for every $P \in \mathrm{CRec}(\Sigma)$.

(ii) *There exists some* $S_{\mathscr{R}} \in \mathrm{Spec}_{\mathscr{R}}$ *such that there is no corresponding* $S_{\mathscr{F}} \in \mathrm{Spec}_{\mathscr{F}}$ *with* (*).

(iii) *However, if* $S_{\mathscr{R}} \in \mathrm{Spec}_{\mathscr{R}}$ *is convex closed, there is some* $S_{\mathscr{F}} \in \mathrm{Spec}_{\mathscr{F}}$ *with* (*).

Proof. (i) Let $\mathscr{P} = \{P \mid \mathscr{F} \models P \text{ sat } S_{\mathscr{F}}\}$. If $\mathscr{P} = \emptyset$, take some $S_{\mathscr{R}} \in \mathrm{Spec}_{\mathscr{R}}$ such that $\mathscr{R} \models P \text{ sat } S_{\mathscr{R}}$ is not satisfied by any $P \in \mathrm{CRec}(\Sigma)$. If $\mathscr{P} \neq \emptyset$, define $S = \bigcup_{P \in \mathscr{P}} \mathscr{F}[\![P]\!]$ and take

$$S_{\mathscr{R}} = \{s, sX \mid s\emptyset \in S \wedge \forall a \in X : sa\emptyset \in S \wedge s\bar{X} \in S\} \cup \{s\uparrow \mid s\uparrow \in S\}.$$

Then $S_{\mathscr{R}} \in \mathrm{Spec}_{\mathscr{R}}$, $S_{\mathscr{R}}$ is convex closed and $\mathscr{C}_g(S_{\mathscr{R}}) = S \subseteq S_{\mathscr{F}}$. Thus $\mathscr{F}[\![P]\!] \subseteq S_{\mathscr{F}}$ iff $\mathscr{F}[\![P]\!] \subseteq S$ (by definition of S) iff $\mathscr{C}_g(\mathscr{R}[\![P]\!]) \subseteq \mathscr{C}_g(S_{\mathscr{R}})$ iff $\mathscr{R}[\![P]\!] \subseteq S_{\mathscr{R}}$ (by Proposition 11.4 and Lemma 11.6). Hence (*) holds.

(ii) Consider $P = a \to \mathsf{stop}$, $Q = b \to \mathsf{stop}$, and $S_{\mathscr{R}} = \mathscr{R}[\![P \text{ or } Q]\!]$. Suppose (*) holds for some $S_{\mathscr{F}} \in \mathrm{Spec}_{\mathscr{F}}$. Because $\mathscr{F}[\![P \,\Box\, Q]\!] \subseteq \mathscr{F}[\![P \text{ or } Q]\!]$ we get $\mathscr{F} \models P \,\Box\, Q \text{ sat } S_{\mathscr{F}}$. But $\mathscr{R} \models P \,\Box\, Q \text{ sat } S_{\mathscr{R}}$ is false. Contradiction.

(iii) Take $S_{\mathscr{F}} = \mathscr{C}_g(S_{\mathscr{R}})$. Then (*) follows as in (i) from Proposition 11.4 and Lemma 11.6. //

Theorem 11.9. *A liveness property* $\mathscr{L}$ *is expressible in* $\mathscr{R}$ *iff* $\mathscr{L}$ *is expressible in* $\mathscr{F}$.

Proof. "**if**": by Proposition 11.8, (i).

"**only if**": Let $\mathscr{L}$ be expressible in $\mathscr{R}$. Due to Proposition 11.8, (iii) it suffices to show that $\mathscr{L}$ is also expressible by a convex closed specification $S \in \mathrm{Spec}_{\mathscr{R}}$. To see this we reexamine the proof of Proposition 10.5. Consider

$$\mathscr{P} = \{P \mid P \text{ satisfies } \mathscr{L}\}.$$

Case 1. $\mathscr{P} = \emptyset$. Then by Proposition 10.5 $S_{\mathscr{R}}(T, L)$ with an arbitrary $(T, L) \in \mathscr{L}$ expresses $\mathscr{L}$. Since every $S_{\mathscr{R}}(T, L)$ is convex closed, we can take $S = S_{\mathscr{R}}(T, L)$.

Case 2. $\mathscr{P} \neq \emptyset$. Then $S = \bigcup_{P \in \mathscr{P}} \mathscr{R}[\![P]\!]$ expresses $\mathscr{L}$ due Proposition 10.5. We show that S is convex closed. Consider $sX \in S$ and $X \subseteq Y \subseteq \{a \mid sa \in S\}$. Since $\{a \mid sa \in S\}$ is finite and, by Proposition 10.5, $\mathscr{P}$ is closed under the nondeterministic operator or, there exists some $P \in \mathscr{P}$ with:

$$sX \in \mathscr{R}[\![P]\!],$$
$$sa \in \mathscr{R}[\![P]\!] \quad \text{whenever } sa \in S.$$

Then there exists a process P' with

$$\mathscr{R}[\![P']\!] = \mathscr{R}[\![P]\!] \cup \{sY\}.$$

We show that $P' \in \mathscr{P}$, i.e. P' satisfies $\mathscr{L}$. Consider a deterministic behaviour D' of P'. If $sY \notin D'$ then D' is a deterministic behaviour of P and there exists some $(T, L) \in \mathscr{L}$ with

$$D' \subseteq S_{\mathscr{R}}(T, L)$$

because $P \in \mathscr{P}$. Otherwise $sY \in D'$. Then

$$D = (D' - \{sY\}) \cup \{sX\}$$

is a deterministic behaviour of P for which again there is some $(T, L) \in \mathscr{L}$ with

$$D \subseteq S_{\mathscr{R}}(T, L).$$

Since $X \subseteq Y$, the same (T, L) yields

$$D' \subseteq S_{\mathscr{R}}(T, L).$$

This shows $P' \in \mathscr{P}$. Thus $sY \in \mathscr{R}[\![P']\!] \subseteq S$, and S is convex closed. //

Reduction

Let us now return to the original question of reducing finite processes to primitive ones. The advantage of $\mathscr{F}$ over $\mathscr{R}$ is here the following algebraic law of $\mathscr{F}$:

$$(+) \qquad (P \,\Box\, b \to Q) \backslash b = (P \,\Box\, Q) \backslash b \text{ or } Q \backslash b.$$

Note that Eq. (2) in Example 11.2 is just an instance of (+). To show that $\mathscr{F}$ admits reduction from $\mathrm{FRec}(\Sigma)$ to $\mathrm{FRec}(\Sigma p)$ we state some auxiliary laws which hold already in $\mathscr{R}$ (and thus in $\mathscr{F}$ by Proposition 11.4).

(i) $\Box$ is commutative and associative; it has a unit stop and a zero div, i.e.

$$P \,\Box\, \mathsf{stop} = P \quad \text{and} \quad P \,\Box\, \mathsf{div} = \mathsf{div}.$$

(ii) $\|_A$ is commutative and has a zero div:

$$P \|_A \mathsf{div} = \mathsf{div}.$$

(iii) $\backslash b$ has div as a zero:

$$\mathsf{div} \backslash b = \mathsf{div}$$

(iv) or is commutative and associative; it admits distribution by $a \to$, $\Box$, $\backslash b$ and $\|_A$, i.e.

$$a \to (P \text{ or } Q) = (a \to P) \text{ or } (a \to Q)$$
$$(P \text{ or } Q) \,\Box\, R = (P \,\Box\, R) \text{ or } (Q \,\Box\, R)$$
$$(P \text{ or } Q) \backslash b = P \backslash b \text{ or } Q \backslash b$$
$$(P \text{ or } Q) \|_A R = (P \|_A R) \text{ or } (Q \|_A R).$$

By these laws it suffices to restrict ourselves to primitive finite processes involving only

$$\mathsf{stop},\ a\rightarrow \text{ and } \Box$$

when proving reducibility of $\|_A$ and $\backslash b$.

Proposition 11.10. *Parallel composition* $\|_A$ *is reducible to* $\mathrm{FRec}(\Sigma p)$ *in* $\mathscr{R}$ *and thus in* $\mathscr{F}$.

Proof. Consider two restricted primitive processes P and Q. We proceed by structural induction. If $P=Q=\mathsf{stop}$ holds, reducibility of $\|_A$ follows from the law

$$\mathsf{stop}\,\|_A\,\mathsf{stop}=\mathsf{stop}$$

in $\mathscr{R}$. Otherwise P and Q can be written as

$$P=\Box_{b\in B} b\rightarrow P_b \quad \text{and} \quad Q=\Box_{c\in C} c\rightarrow P_c$$

with $B, C\subseteq\mathrm{Comm}$. If P or Q is stop, we choose B or C to be empty. Reducibility of $\|_A$ follows from the induction hypothesis and the law

$$P\|_A Q=(\Box_{b\in B-A} b\rightarrow(P_b\|_A Q))\Box(\Box_{c\in C-A} c\rightarrow(P\|_A Q_c))$$
$$\Box(\Box_{b=c\in A\cap B\cap C} b\rightarrow(P_b\|_A Q_c))$$

in $\mathscr{R}$. //

Proposition 11.11. *Hiding* $\backslash b$ *is reducible to* $\mathrm{FRec}(\Sigma p)$ *in* $\mathscr{F}$.

Proof. By structural induction. For $P=\mathsf{stop}$ reducibility of $\backslash b$ follows from the law

$$\mathsf{stop}\backslash b=\mathsf{stop}$$

in ($\mathscr{R}$ and) $\mathscr{F}$. Otherwise P is of the form

$$P=\Box_{a\in A} a\rightarrow P_a$$

with $A\subseteq\mathrm{Comm}$. Reducibility of $\backslash b$ follows then from the induction hypothesis and the following case analysis. If $b\notin A$ then

$$P\backslash b=\Box_{a\in A} a\rightarrow(P_a\backslash b)$$

holds in ($\mathscr{R}$ and) $\mathscr{F}$. If $b\in A$ we apply law (+) above which is valid only in $\mathscr{F}$. //

The previous propositions are summarised in:

Corollary 11.12. *The Failure Model* $\mathscr{F}$ *admits reduction from* $\mathrm{FRec}(\Sigma)$ *to* $\mathrm{FRec}(\Sigma p)$.

Discussion

The Failure Model originally proposed in [13] has recently attracted much attention in the literature. Whereas our Failure Model $\mathscr{F}$ can be considered as a refinement of the Divergence Model $\mathscr{D}$, the original model in [13] is a refinement of the Trace Model $\mathscr{T}$ discussed in Sect. 7. Consequently the problems concerning divergence signalled in Sect. 8 are also present in the original model [13]. This was first realised independently in work of [11, 39, 46].

Our present model $\mathscr{F}$, which takes divergence into account, is isomorphic to the Improved Failure Model described in [14]. In [14] also a complete proof system for the semantic equality of finite processes is given. This proof system incorporates the algebraic laws needed here to show the reduction property of $\mathscr{F}$.

The Failure Model $\mathscr{F}$ is closely related to a model SRT of [40] developed in the context of Milner's calculus CCS [34]. DeNicola and Hennessy [40] start from a general idea of *testing processes*. Two processes P and Q are considered equivalent if and only if they pass exactly the same tests. Roughly, a test T is itself a process, and testing another process P means running P and T in parallel. DeNicola and Hennessy consider three definitions of "passing a test", each one leading to a different notion of equivalence on processes. Their second equivalence $\cong_2$ generates the model SRT which uses so-called *acceptance sets* (see also [23]).

Translated into the framework of Communicating Processes, SRT can be viewed as applying the convex closure to our Readiness Model $\mathscr{R}$:

$$\mathrm{SRT}[\![P]\!] = \mathrm{con}(\mathscr{R}[\![P]\!]).$$

Then Corollary 11.7 implies that

$$\mathrm{SRT}[\![P]\!] = \mathrm{SRT}[\![Q]\!] \quad \text{iff } \mathscr{F}[\![P]\!] = \mathscr{F}[\![Q]\!]$$

(see also [23], Conclusion). Thus DeNicola and Hennessy's notion of testing can very well serve as a motivation for considering the Failure Model $\mathscr{F}$.

Another characterisation of $\mathscr{F}$ for the subset of finite processes without divergence and hiding is reported in [9]. It is shown that failure equivalence is the maximal congruence which respects complete traces. In other words,

$$\mathscr{F}[\![P]\!] = \mathscr{F}[\![Q]\!]$$

if and only if in all contexts $\mathscr{C}(\xi)$

$$\mathscr{T}_c[\![\mathscr{C}(P)]\!] = \mathscr{T}_c[\![\mathscr{C}(Q)]\!].$$

By definition, a context is a term $\mathscr{C}(\xi) \in \mathrm{Rec}(\Sigma)$ with a free identifier ξ for which P and Q are substituted, and $\mathscr{T}_c[\![P]\!]$ denotes the set of complete traces of a process P given by

$$\mathscr{T}_c[\![P]\!] = \{s \in \mathrm{Comm}^* \mid \text{where } s\emptyset \in \mathscr{R}[\![P]\!]\}.$$

Thus for the processes considered in [9] the idea of failures can be explained simply in terms of complete traces and contexts.

In our paper we have presented the Failure Model $\mathscr{F}$ as a special example in the general setting of specification-oriented semantics. Together with the series of models $\mathscr{C}$, $\mathscr{T}$, $\mathscr{D}$ and $\mathscr{R}$ we hope this gives a better insight into the structure of $\mathscr{F}$ and its relationship to the other models.

12. Operational Semantics

In the previous sections we studied a series of denotational models for Communicating Processes. But every now and then we appealed to some "operational" intuitions about processes in order to motivate particular notions, for example the notion of "divergence" in Sect. 8 and of "stability" in Sect. 10. It seems therefore appropriate to make these operational intuitions precise and relate them with our models.

To do so we follow Milner and use the concept of transitions [32, 34, 44]. The advantage of transitions is that an explicit symbol τ denoting an internal action allows simple definitions. The drawback is of course that we lose abstraction from internal activity - the main concern in our specification-oriented approach. We thus start from a set

$$(\lambda \in)\,\mathrm{Act} = \mathrm{Comm} \cup \{\tau\}$$

of *actions*. An action λ is either an observable communication $a \in \mathrm{Comm}$ or the internal action τ. *Transitions* or *rewriting rules* are binary relations $\xrightarrow{\lambda}$ over $\mathrm{CRec}(\Sigma)$ with $\lambda \in \mathrm{Act}$. Informally

$$P \xrightarrow{\lambda} Q$$

means that P can first do action λ and then behave like Q. In particular $P \xrightarrow{\tau} Q$ means that P can transform itself into Q without communication to its environment.

For $\lambda \in \mathrm{Act}$ let $\xrightarrow{\lambda}$ be the smallest relation over $\mathrm{CRec}(\Sigma)$ with:

(1) stop has no transition.

(2) $\mathsf{div} \xrightarrow{\tau} \mathsf{div}$

(3) $(a \to P) \xrightarrow{a} P$

(4) $(P \text{ or } Q) \xrightarrow{\tau} P$ and $(P \text{ or } Q) \xrightarrow{\tau} Q$

(5) If $P \xrightarrow{a} P1$ then $(P \,\square\, Q) \xrightarrow{a} P1$ and $(Q \,\square\, P) \xrightarrow{a} P1$.
If $P \xrightarrow{\tau} P1$ then $(P \,\square\, Q) \xrightarrow{\tau} (P1 \,\square\, Q)$ and $(Q \,\square\, P) \xrightarrow{\tau} (Q \,\square\, P1)$.

Only the first observable communication a decides whether $P \,\square\, Q$ behaves like P or like Q. As long as one of its components P or Q pursues internal actions τ, the process $P \,\square\, Q$ does not withdraw the option of selecting the other component. This implicit abstraction from internal actions τ is the essential difference between $\square$ and Milner's operator $+$ which satisfies for all $\lambda \in \mathrm{Act}$:

If $P \xrightarrow{\lambda} P1$ then $(P + Q) \xrightarrow{\lambda} P1$ and $(Q + P) \xrightarrow{\lambda} P1$ [34].

Unfortunately, this simple definition of + causes difficulties expressed by the fact that Milner's observational equivalence is not a congruence (see [34], Chap. 7). These difficulties are avoided by taking $\Box$ instead of +. Further details on $\Box$ and + are given in the Examples 12.1 and 12.3 below.

(6) If $a \in A$ and $P \xrightarrow{a} P1$, $Q \xrightarrow{a} Q1$ then $P \|_A Q \xrightarrow{a} P1 \|_A Q1$.
If $\lambda \notin A$ and $P \xrightarrow{\lambda} P1$ then $P \|_A Q \xrightarrow{\lambda} P1 \|_A Q$ and $Q \|_A P \xrightarrow{\lambda} Q \|_A P1$.

(7) If $P \xrightarrow{b} Q$ then $P \backslash b \xrightarrow{\tau} Q \backslash b$.
If $\lambda \neq b$ and $P \xrightarrow{\lambda} Q$ then $P \backslash b \xrightarrow{\lambda} Q \backslash b$.

(8) $\mu\xi \cdot P \xrightarrow{\tau} P[\mu\xi \cdot P/\xi]$.

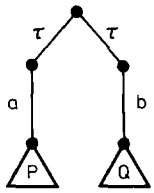
Recursion is modelled by the *copy rule* known from procedural languages such as ALGOL. Copying is done here as an internal action.

It is sometimes helpful to visualise the possible transitions of a process by so-called *synchronisation trees* [34]. These are rooted, unordered trees whose arcs are labelled with actions $\lambda \in$ Act. We show some typical cases.

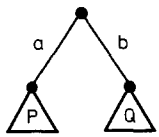
Example 12.1.

(i) div =

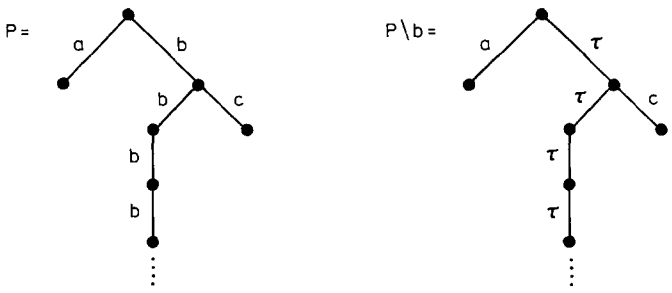
(ii) $(a \to P)$ or $(b \to Q) =$



(iii) $(a \to P) \Box (b \to Q) =$

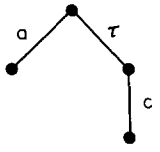


(iv) Hiding b in a synchronisation tree P simply means relabelling all arcs b into τ:



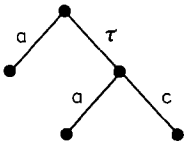
(v) Combining external nondeterminism $\square$ and hiding is rather delicate. Let $P = a \rightarrow \mathsf{stop}$ and $Q = b \rightarrow c \rightarrow \mathsf{stop}$. Then

$(P \square Q)\backslash b =$



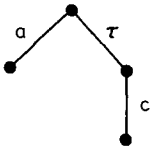
analogously to (iv) but

$P \square (Q\backslash b) =$



since here the τ-action of $Q\backslash b$ does not remove the communication a of P. For Milner's $+$ we would obtain identical trees:

$(P+Q)\backslash b =$



$= P + (Q\backslash b)$. //

Transitions $\xrightarrow{\lambda}$ are extended in two ways. For words $w = \lambda_1 \ldots \lambda_n \in \mathrm{Act}^*$ let $\xrightarrow{w}$ be the relational product

$$\xrightarrow{w} = \xrightarrow{\lambda_1} \circ \ldots \circ \xrightarrow{\lambda_n}$$

of the individual transitions $\xrightarrow{\lambda_i}$, and for traces $s \in \mathrm{Comm}^*$ we write

$$P \stackrel{s}{\Rightarrow} Q$$

if there exists some $w \in \mathrm{Act}^*$ with $P \xrightarrow{w} Q$ and $s = w\backslash\tau$ where $w\backslash\tau$ denote the result of removing all occurrences of τ inside w.

We can now define the important concept of divergence in an operational setting. A process P *diverges at* s if

$$\exists Q\, \forall n \geqq 0\, \exists R\colon\ P \stackrel{s}{\Rightarrow} Q \wedge Q \xrightarrow{\tau^n} R.$$

P is *divergence free* if there is no s at which P diverges. (As we shall see in the next section, this operational definition agrees with the earlier Definition 8.3.)

Example 12.2. (i) $a \rightarrow \mathsf{div}$ diverges at a.

(ii) div, $\mu\xi \cdot \xi$ and $(\mu\xi \cdot b \rightarrow \xi)\backslash b$ all diverge at ε. //

Finally, we introduce a modification of Milner's *observational equivalence* $\approx$ [24, 34, 35] which takes the notion of divergence into account. $\approx$ is defined by the following series of equivalence relations $\approx_l$, $l \geqq 0$, over $\mathrm{CRec}(\Sigma)$:

$P \approx_0 Q$ if either both P and Q diverge at ε
or both P and Q don't diverge at ε.

$P \approx_{l+1} Q$ if $\forall s \in \mathrm{Comm}^*$ with $|s| \leqq 1$:
(i) $P \overset{s}{\Rightarrow} P1$ implies $\exists Q1\colon\ Q \overset{s}{\Rightarrow} Q1 \wedge P1 \approx_l Q1$
(ii) $Q \overset{s}{\Rightarrow} Q1$ implies $\exists P1\colon\ P \overset{s}{\Rightarrow} P1 \wedge Q1 \approx_l Q1$

$P \approx Q$ if $P \approx_l Q$ holds for all $l \geqq 0$.

In contrast, Milner's original definition of observational equivalence in [34] ignores divergence and puts

$$P \approx_0 Q$$

to be universally true. Intuitively, checking $P \approx_l Q$ means investigating the synchronisation trees of P and Q along all branches

$$w = \lambda_1 \ldots \lambda_n$$

with at most l observable communications $\lambda_i \in \mathrm{Comm}$. Since this does not exclude branches with arbitrarily many internal actions τ, it is in general impossible to establish $P \approx_l Q$ effectively.

Example 12.3. (i) $(b \to P) \backslash b \approx P \backslash b$.

(ii) $((b \to P) \backslash b) \,\square\, Q \approx (P \backslash b) \,\square\, Q$.

(iii) $\mathsf{div} \approx \mu\xi \cdot \xi$, but $\mathsf{div} \not\approx_1 \mathsf{stop}$ or div.

(iv) $a \to (P \text{ or } Q) \not\approx_2 (a \to P)$ or $(a \to Q)$.

Clause (ii) does not hold for Milner's $+$:

$$((b \to P) \backslash b) + Q \not\approx (P \backslash b) + Q$$

in general. Together with (i) this rephrases Milner's observation that $\approx$ is not a congruence w.r.t. $+$. We remark without proof that $\approx$ is a congruence w.r.t. $\square$.

Note that the example exhibits differences between the algebraic laws in the previous denotational models and the operationally defined observational equivalence: in contrast to (iii) and (iv) the laws

$$\mathsf{div} = \mathsf{stop} \text{ or } \mathsf{div},$$
$$a \to (P \text{ or } Q) = (a \to P) \text{ or } (a \to Q)]$$

hold in all models $\mathscr{C}$, $\mathscr{T}$, $\mathscr{D}$, $\mathscr{R}$ and $\mathscr{F}$ (cf. Sect. 8 and 11). The precise relationship between the denotational models and the operational semantics is the topic of the next section.

13. Consistency

To relate our specification-oriented models with the operational transition semantics we now add a logical structure $\Vdash$ to observations which explains

how we actually make observations about processes. $\Vdash$ is defined as a relation between processes P and observations x. We write

$$P \Vdash x$$

and say that x is a *possible observation* about P. We shall require that $\Vdash$ "agrees" with the observational equivalence $\approx$ introduced in the previous section. Recall from Sect. 4 that the level $\|x\|$ of an observation x is the minimum length of a grounded chain for x, measuring the observable progress a process has to make before the observation x is possible.

Definition 13.1. A *logical structure* for Obs is a relation $\Vdash \subseteq \mathrm{CRec}(\Sigma) \times \mathrm{Obs}$ such that for any $l \geqq 0$, any observation $x \in \mathrm{Obs}$ with level $\|x\| = l$ and any two processes $P, Q \in \mathrm{CRec}(\Sigma)$ with $P \approx_l Q$

$$P \Vdash x \quad \text{iff } Q \Vdash x$$

holds. //

Informally, this definition says that observations are finitary and abstract. *Finitary* in the sense that an observation x investigates a process P only up to the equivalence $\approx_l$ where $l = \|x\|$. (By the definition of $\approx_l$ this does not imply the effectiveness of $\Vdash$.) And *abstract* in the sense that due to $\approx$ finite linear branches of internal action τ in synchronisation trees of processes P are not detectable by observations: cf. Example 12.2, (i) and (ii).

We can now be precise about the desired relationship between models $\mathscr{M}$ and the transition semantics.

Definition 13.2. Let $\Sigma 0 \subseteq \Sigma$ and $\mathscr{M}$ be a specification-oriented model for $\mathrm{CRec}(\Sigma 0)$ over $(\mathrm{Obs}, \longrightarrow, \Longrightarrow)$. Then $\mathscr{M}$ is called (*weakly*) *consistent* if there exists a logical structure $\Vdash$ for Obs such that

$$\mathscr{M}[\![P]\!] = \{x \in \mathrm{Obs} \mid P \Vdash x\}$$

holds for every (divergence free) process $P \in \mathrm{CRec}(\Sigma 0)$. More precisely we say that $\mathscr{M}$ is (weakly) consistent *w.r.t.* $\Vdash$. //

Informally, $\mathscr{M}$ exactly computes the set of observations we can make about P. If $\Vdash$ is known, we abbreviate

$$\mathrm{Obs}[\![P]\!] = \{x \in \mathrm{Obs} \mid P \Vdash x\}.$$

Next we state a general theorem which simplifies the task of checking the consistency of a model $\mathscr{M}$.

Theorem 13.3. *Let $\Sigma 0 \subseteq \Sigma$, $\mathscr{M}$ be a specification-oriented model for* $\mathrm{CRec}(\Sigma 0)$ *over* $(\mathrm{Obs}, \longrightarrow, \Longrightarrow)$, *and $\Vdash$ be a logical structure for* Obs. *Suppose that for all processes $f(P_1, \ldots, P_n) \in \mathrm{CRec}(\Sigma 0)$ the following holds:*

(1) $\mathrm{Obs}[\![f(P_1, \ldots, P_n)]\!] = f_{\mathscr{M}}(\mathrm{Obs}[\![P_1]\!], \ldots, \mathrm{Obs}[\![P_n]\!])$,

(2) $\mathrm{Obs}[\![P]\!] = \mathrm{Obs}$ *whenever P diverges at ε.*

Then $\mathscr{M}$ is consistent w.r.t. $\Vdash$.

Proof. Clearly $\mathcal{M}[\![P]\!] = \mathrm{Obs}[\![P]\!]$ holds for all finite, i.e. non-recursive processes P due to (1). But some care is needed to show that the copy rule definition of recursion $\mu\xi \cdot P$ agrees with the fixed point definition in $\mathcal{M}$. To this end, we consider three types of assertions:

$$A_m\colon\ \mathcal{M}[\![P]\!] = \mathrm{Obs}[\![P]\!]$$

holds for every $P \in \mathrm{CRec}(\Sigma 0)$ with at most m occurrences of μ.

$$B_m\colon\ \mathrm{Obs}[\![P(\bar{Q})]\!] \subseteq \mathrm{Obs}[\![P(\bar{R})]\!]$$

holds for every $P \in \mathrm{Rec}(\Sigma)$ with at most m occurrences of μ and at most r free identifiers $\xi_1, \ldots, \xi_r$ for which lists $\bar{Q} = Q_1, \ldots, Q_r$ and $\bar{R} = R_1, \ldots, R_r$ of processes in $\mathrm{CRec}(\Sigma 0)$ with $\mathrm{Obs}[\![Q_i]\!] \subseteq \mathrm{Obs}[\![R_i]\!]$ for $i = 1, \ldots, r$ are substituted.

$$C_m\colon\ \mathrm{Obs}[\![\mu\xi \cdot P(\bar{Q})]\!] = \bigcap_{n \geqq 0} \mathrm{Obs}[\![(P(\bar{Q}))^n(\mathsf{div})]\!]$$

holds for every $P \in \mathrm{Rec}(\Sigma)$ with at most m occurrences of μ and at most $r+1$ free identifiers $\xi_1, \ldots, \xi_r$ and ξ for which a list $\bar{Q} = Q_1, \ldots, Q_r$ of processes in $\mathrm{CRec}(\Sigma 0)$ is substituted to yield $P(\bar{Q})$ with at most ξ as free identifier. To ξ an n-fold substitution starting with div is applied to yield $(P(\bar{Q}))^n(\mathsf{div})$.

We wish to show that A_m holds for every $m \geqq 0$. But to do this we will use the B_m's and C_m's as well.

Clearly A_0 and B_0 are true because of (1) and the monotonicity of the operators in $\mathcal{M}$. Using (1) and the continuity of the operators in $\mathcal{M}$ it is easy to see that

$$C_m \text{ and } A_m \text{ together imply } A_{m+1},$$
$$C_m \text{ and } B_m \text{ together imply } B_{m+1}.$$

Thus to show A_m, B_m, C_m for every $m \geqq 0$, it suffices to prove

$$B_m \text{ implies } C_m.$$

So choose some $m \geqq 0$ and assume B_m. Let P abbreviate $P(\bar{Q})$.

Case 1. $\mu\xi \cdot P$ diverges at ε.

Then $P^n(\mathsf{div})$ diverges at ε for every $n \geqq 0$. Thus C_m follows from (2).

Case 2. $\mu\xi \cdot P$ does not diverge at ε.

Since $\mu\xi \cdot P \xrightarrow{\tau} P(\mu\xi \cdot P)$ is the only initial transition that $\mu\xi \cdot P$ can perform, we get $\mu\xi \cdot P \approx P(\mu\xi \cdot P)$ and thus $\mathrm{Obs}[\![\mu\xi \cdot P]\!] = \mathrm{Obs}[\![P(\mu\xi \cdot P)]\!]$. B_m implies that in fact

$$(*) \qquad \mathrm{Obs}[\![\mu\xi \cdot P]\!] = \mathrm{Obs}[\![P^n(\mu\xi \cdot P)]\!]$$

holds for every $n \geqq 0$. For $S \subseteq \mathrm{Obs}$ and $l \geqq 0$ we define

$$l\colon S = \{x \in S \mid \text{where } \|x\| = l\}.$$

We show that

$$(**) \qquad l\colon \mathrm{Obs}[\![P^n(R_1)]\!] = l\colon \mathrm{Obs}[\![P^n(R_2)]\!]$$

holds for every $n \geqq l+1$ and all $R_1, R_2 \in \mathrm{CRec}(\Sigma)$:

Take some $R \in \mathrm{CRec}(\Sigma)$. R is called *guarded* in $P(R)$ if there exist $G, H_i \in \mathrm{Rec}(\Sigma)$ and $b_i \in \mathrm{Comm}$ such that $P(R)$ is of the syntactic form

$$P(R) = G(b_1 \to H_1(R), \ldots, b_n \to H_n(R)),$$

and no further R occurs in G.

Since $\mu\xi \cdot P$ does not diverge at ε, there is no transition chain

$$P(R) \xrightarrow{\lambda_1 \ldots \lambda_k} P^*(R)$$

where all $\lambda_i = \tau$ and R is not guarded in $P^*(R)$. Thus at least *one* λ_i is an observable communication, say $\lambda_i = b \in \mathrm{Comm}$. Consequently, every transition chain

$$P^n(R) \xrightarrow{\lambda_1 \ldots \lambda_k} P^{**}(R)$$

such that R is not guarded in $P^{**}(R)$ needs at least n observable $\lambda_i \in \mathrm{Comm}$. Hence $P^n(R_1) \approx_l P^n(R_2)$ holds for all $R_1, R_2 \in \mathrm{CRec}(\Sigma)$. This implies (**).

We can now verify C_m:

$$\begin{aligned}
\mathrm{Obs}[\![\mu\xi \cdot P]\!] &= \bigcup_{l \geqq 0} l\colon \mathrm{Obs}[\![\mu\xi \cdot P]\!] && \text{(definition } l\colon S\text{)} \\
&= \bigcup_{l \geqq 0} l\colon \Big(\bigcap_{n \geqq l+1} \mathrm{Obs}[\![P^n(\mu\xi \cdot P)]\!] \Big) && \text{(by (*))} \\
&= \bigcup_{l \geqq 0} \Big(\bigcap_{n \geqq l+1} l\colon \mathrm{Obs}[\![P^n(\mu\xi \cdot P)]\!] \Big) && \text{(definition } l\colon S\text{)} \\
&= \bigcup_{l \geqq 0} \Big(\bigcap_{n \geqq l+1} l\colon \mathrm{Obs}[\![P^n(\mathsf{div})]\!] \Big) && \text{(by (**))} \\
&= \bigcup_{l \geqq 0} l\colon \Big(\bigcap_{n \geqq l+1} \mathrm{Obs}[\![P^n(\mathsf{div})]\!] \Big) && \text{(definition } l\colon S\text{)} \\
&= \bigcup_{l \geqq 0} l\colon \Big(\bigcap_{n \geqq 0} \mathrm{Obs}[\![P^n(\mathsf{div})]\!] \Big) && \text{(by } B_m \text{ and (2))} \\
&= \bigcap_{n \geqq 0} \mathrm{Obs}[\![P^n(\mathsf{div})]\!] && \text{(definition } l\colon S\text{).}
\end{aligned}$$

This finishes our proof. //

We apply now Theorem 13.3 to relate our most detailed model, the Readiness Model $\mathscr{R}$, with the transition semantics. First we need two auxiliary notions.

Definition 13.4. For a process P the set of *next possible actions* is given by

$$\mathrm{next}(P) = \{\lambda \mid \exists Q\colon P \xrightarrow{\lambda} Q\}.$$

We call P *stable* if $\tau \notin \mathrm{next}(P)$. A *stable state* of P is a stable process R with $P \overset{t}{\Longrightarrow} R$ for some trace $t \in \mathrm{Comm}^*$. //

This explains the notion of stability used earlier in Sect. 10. Now we can define a logical structure $\Vdash_{\mathscr{R}}$ for $\mathrm{Obs}_{\mathscr{R}}$:

$P \Vdash_{\mathscr{R}} t\uparrow$ iff $\exists s \leqq t$: P diverges at s.

$P \Vdash_{\mathscr{R}} t$ iff $\exists Q$: $P \overset{t}{\Rightarrow} Q \vee P \Vdash_{\mathscr{R}} t\uparrow$

$P \Vdash_{\mathscr{R}} tX$ iff $(\exists Q$: $P \overset{t}{\Rightarrow} Q \wedge Q$ stable $\wedge$ next $(Q) = X) \vee (P \Vdash_{\mathscr{R}} t\uparrow)$.

Proposition 13.5. *The Readiness Model $\mathscr{R}$ is consistent w.r.t. $\Vdash_{\mathscr{R}}$.*

Proof. Condition (2) of Theorem 13.3 holds by the definition of $\Vdash_{\mathscr{R}}$. For condition (1) let us present the crucial facts for two of the more interesting cases: $\Box$ and $\backslash b$.

(a) Crucial for proving

$$\mathrm{Obs}_{\mathscr{R}}[\![P \Box Q]\!] = \mathrm{Obs}_{\mathscr{R}}[\![P]\!] \Box_{\mathscr{R}} \mathrm{Obs}_{\mathscr{R}}[\![Q]\!]$$

are the following two assertions:

(i) $P \Box Q$ diverges at s iff P diverges at s or Q diverges at s.

(ii) $P \Box Q \overset{\varepsilon}{\Rightarrow} R \wedge R$ stable $\wedge$ next $(R) = X$

iff $\exists R_1, R_2, X_1, X_2$: $P \overset{\varepsilon}{\Rightarrow} R_1 \wedge Q \overset{\varepsilon}{\Rightarrow} R_2 \wedge R_1, R_2$ stable

$\wedge$ next $(R_1) = X_1 \wedge$ next $(R_2) = X_2 \wedge R = R_1 \Box R_2 \wedge X = X_1 \cup X_2$

(b) For establishing

$$\mathrm{Obs}_{\mathscr{R}}[\![P \backslash b]\!] = \mathrm{Obs}_{\mathscr{R}}[\![P]\!] \backslash_{\mathscr{R}} b$$

we need the following:

(i) $P \backslash b$ diverges at t

iff $\exists s\, \exists Q$: $s \backslash b = t \wedge P \overset{s}{\Rightarrow} Q$

$\wedge (Q$ diverges at $\varepsilon \vee \forall n \geqq 0\, \exists R$: $Q \overset{b^n}{\Rightarrow} R)$.

(ii) $P \backslash b \overset{t}{\Rightarrow} R \wedge R$ stable $\wedge$ next $(R) = X$

iff $\exists s\, \exists Q$: $s \backslash b = t \wedge Q \backslash b = R \wedge P \overset{s}{\Rightarrow} Q$

$\wedge\, Q$ stable $\wedge\, b \notin$ next $(Q) \wedge$ next $(Q) = X$.

Then we get

$$\mathrm{Obs}_{\mathscr{R}}[\![P \backslash b]\!] = \{s \backslash b, (s \backslash b)X \mid sX \in \mathrm{Obs}_{\mathscr{R}}[\![P]\!] \wedge b \notin X\}$$
$$\{(s \backslash b)t \triangle \mid s\uparrow \in \mathrm{Obs}_{\mathscr{R}}[\![P]\!] \vee \forall n \geqq 0\colon sb^n \in \mathrm{Obs}_{\mathscr{R}}[\![P]\!]\}.$$

The disjunct "$s\uparrow \in \mathrm{Obs}_{\mathscr{R}}[\![P]\!]$" can be removed since it implies $\mathrm{Obs}_{\mathscr{R}}[\![P]\!] = \mathrm{Obs}_{\mathscr{R}}$ (by condition (2)) and thus also $\forall n \geqq 0$: $sb^n \in \mathrm{Obs}_{\mathscr{R}}[\![P]\!]$. Hence (b) follows.

The remaining operators require similar arguments. //

(Weak) consistency of the more abstract models can be stated as corollaries of Proposition 13.5.

Corollary 13.6. *The Failure Model $\mathscr{F}$ and the Divergence Model $\mathscr{D}$ are both consistent.*

Proof. Propositions 11.4 and 10.7 explain how to modify $\Vdash_{\mathscr{R}}$ to obtain consistent logical structures $\Vdash_{\mathscr{F}}$ and $\Vdash_{\mathscr{D}}$ for $\mathscr{F}$ and $\mathscr{D}$. //

The Trace Model $\mathscr{T}$, however, is not fully consistent with the transition semantics. Already when introducing the Divergence Model $\mathscr{D}$ in Sect. 8 we argued that the law

$$(*) \qquad \mathsf{div} \|_{\mathrm{Comm}} P = P$$

of $\mathscr{T}$ looks unrealistic. Indeed (*) is the reason for $\mathscr{T}$'s inconsistency. To see this look at the example $P = \mathsf{stop}$. Since

$$\mathsf{div} \|_{\mathrm{Comm}} \mathsf{stop} \xrightarrow{\tau} \mathsf{div} \|_{\mathrm{Comm}} \mathsf{stop}$$

is the only transition of $\mathsf{div} \|_{\mathrm{Comm}} \mathsf{stop}$, no logical structure $\Vdash$ can distinguish between $\mathsf{div} \|_{\mathrm{Comm}} \mathsf{stop}$ and div. Thus in every consistent model $\mathscr{M}$ the law

$$\mathsf{div} \|_{\mathrm{Comm}} \mathsf{stop} = \mathsf{div}$$

holds. But in $\mathscr{T}$ this law is false due to (*). (An analogous argument applies for the Counter Model $\mathscr{C}$.)

Nevertheless we can state:

Corollary 13.7. *The Trace Model $\mathscr{T}$ and the Counter Model $\mathscr{C}$ are weakly consistent.*

Proof. By Theorem 8.4 we can choose for $\mathscr{T}$ the following logical structure $\Vdash_{\mathscr{T}}$:

$$P \Vdash_{\mathscr{T}} s \quad \text{iff } \exists Q\colon P \overset{s}{\Rightarrow} Q.$$

The logical structure $\Vdash_{\mathscr{C}}$ for $\mathscr{C}$ is then clear from Proposition 7.1. //

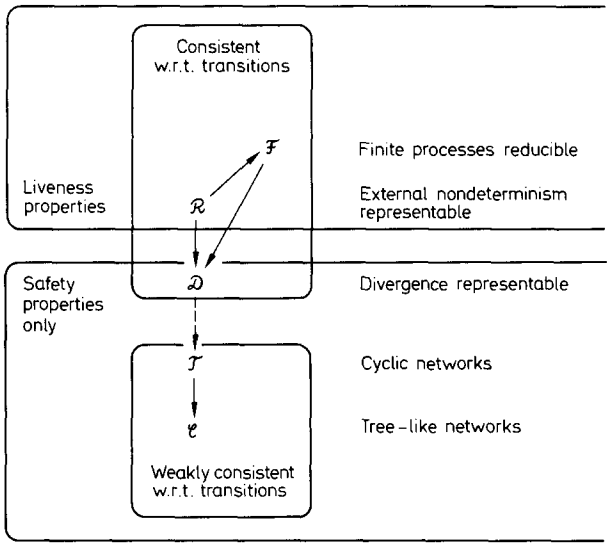
14. Conclusion

Starting from a simple idea of process correctness we developed a specific form of denotational semantics for processes, called specification-oriented semantics. This approach provided a uniform framework for discussing a series of increasingly sophisticated models for Communicating Processes in a step-by-step manner. Our results are summarised in Diagram 1 where arrows $\rightarrow$ ($\dashrightarrow$) denote (weak) homomorphisms.

Diagram 1 explains the purposes and applications for which these models are best suited. For example, if we wish to reason about safety in divergence free cyclic networks of processes, we don't need the complex Failure Model $\mathscr{F}$; it suffices to choose the Trace Model $\mathscr{T}$. Also the models can be combined to new ones. For example, for reasoning about liveness properties in acyclic networks a simplified readiness model with communication counters instead of traces would do.

All our models were based on the idea of finite observations. Other authors find the concept of infinite streams more natural (e.g. [4, 15]). One might ask whether there is any interesting relationship between theses approaches. In fact there is: in [5] an isomorphism is established between the stream-based semantics of [4] and an observation-based semantics situated "in between" the

Diagram 1



Divergence Model $\mathscr{D}$ and the Readiness Model $\mathscr{R}$. This isomorphism has benefits for both approaches (see [5]).

A notable omission in our programming language is the notion of state. This would allow to add assignment and explicit value passing between processes, thus combining sequential programs with Communicating Processes. We have not yet investigated such an addition to our formal framework. But it is clear that some care is needed since the set of states is usually infinite. For example, we would have to consider observation spaces where the extensibility relation $\multimap$ is not image finite any more. Fortunely, such a change does not effect our continuity results in Sect. 5.

In general, it would be interesting to establish some formal relationship between our idea of observations and the more basic concept of events in computation [51].

Also an explicit syntax for specifications and direct proof systems for the relation P sat S should be developed. This could well be done along the lines of [16, 27, 37, 52]. An advantage of starting from one of the models $\mathscr{C}$ to $\mathscr{F}$ would be that the question of completeness of the resulting proof system could be answered more transparently [1, 2, 33, 49].

Perhaps even more important, we hope that our investigations of semantical models for Communicating Processes will provide a firm basis for a mathematical style of programming which allows a free mixture of programming constructs and specifications expressed as predicates [21]. That this style can support a systematic development of concurrent programs from their specifications is demonstrated in [41].

Acknowledgements. Preliminary versions of this paper were presented at various seminars in Oxford, Leeds, Altenahr, Bad Honnef, Edinburgh, Yorktown Heights, Venice and Barcelona. Criticism and suggestions at these meetings helped to bring the paper into its present form. In

particular, we wish to thank M. Broy, C. Crasemann, L. Czaja, M.G. Gouda, M. Hennessy, He Jifeng, G. Jones, R. Milner, G. Niklasch, W.P. de Roever and A.W. Roscoe. The first author was supported by the German Research Council (DFG) under grant No. La426/3-1, by the University of Kiel in granting him leave of absence to join the Programming Research Group at Oxford, by Oxford University in providing further support, and by Wolfson College at Oxford in providing a most enjoyable atmosphere for meeting new friends.

References

1. Apt, K.R., Francez, N., de Roever, W.P.: A proof system for communicating sequential processes. ACM TOPLAS **2**, 359–385 (1980)
2. Apt, K.R.: Formal justification of a proof system for communicating sequential processes. J. Assoc. Comput. Mach. **30**, 197–216 (1983)
3. de Bakker, J.W.: Mathematical theory of program correctness. London: Prentice Hall 1980
4. de Bakker, J.W., Bergstra, J.A., Klop, J.W., Meyer, J.-J.C.: Linear time and branching time semantics for recursion with merge. TCS **34**, 134–156 (1984)
5. de Bakker, J.W., Meyer, J.-J.C., Olderog, E.-R.: Infinite streams and finite observations in the semantics of uniform concurrency. In: Proc. 12th Coll. Automata, Languages and Programming. (W. Brauer ed.). Lect. Notes Comput. Sci. **194**, 149–157 (1985)
6. de Bakker, J.W., Meyer, J.-J.C., Olderog, E.-R., Zucker, J.I.: Transition systems, infinitary languages and the semantics of uniform concurrency. In: Proc. 17th ACM Symposium on Theory of Computing, pp. 252–262. Providence 1985
7. de Bakker, J.W., Zucker, J.I.: Processes and the denotational semantics of concurrency. Inf. Control **54**, 70–120 (1982)
8. Bergstra, J.A., Klop, J.W.: Algebra of communicating processes with abstraction. TCS **37**, 77–121 (1985)
9. Bergstra, J.A., Klop, J.W., Olderog, E.-R.: Readies and failures in the algebra of communicating processes. Report CS-R8523, Center for Mathematics and Computer Science, Amsterdam 1985
10. Brock, J.D., Ackermann, W.B.: Scenarios: a model for nondeterminate computations. In: Formalisation of Programming Concepts (J. Diaz, I. Ramos, eds.). Lect. Notes Comput. Sci. **107**, 252–267 (1981)
11. Brookes, S.D.: A model for communicating sequential processes. D. Phil. Thesis, Oxford Univ. 1983
12. Brookes, S.D.: On the relationship of CCS and CSP. In: Proc. 10th Coll. Automata, Languages and Programming (J. Diaz, ed.). Lect. Notes Comput. Sci. **154**, 83–96 (1983)
13. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. J. Assoc. Comput. Mach. **31**, 560–599 (1984)
14. Brookes, S.D., Roscoe, A.W.: An improved failures model for communicating sequential processes. In: Proc. NSF-SERC Seminar on Concurrency. Lect. Notes Comput. Sci. **197**, 281–305 (1985)
15. Broy, M.: Fixed point theory for communication and concurrency. In: Formal Description of Programming Concepts II (D. Bjørner, ed.), pp. 125–146. Amsterdam: North Holland 1983
16. Chaochen, Z., Hoare, C.A.R.: Partial correctness of communicating processes. In: Proc. 2nd International Conference on Distributed Computing Systems. Paris 1981
17. Francez, N., Hoare, C.A.R., Lehmann, D.J., de Roever, W.P.: Semantics of nondeterminism, concurrency and communication. J. Comput. Syst. Sci. **19**, 290–308 (1979)
18. Francez, N., Lehmann, D., Pnueli, A.: A linear history semantics for languages for distributed programming. TCS **32**, 25–46 (1984)
19. Goguen, J.A., Thatcher, J.W., Wagner, E.G., Wright, J.B.: Initial algebra semantics and continuous algebras. J. Assoc. Comput. Mach. **24**, 68–95 (1977)
20. Guessarian, I.: Algebraic semantics. Lect. Notes Comput. Sci. **99** (1981)
21. Hehner, E.C.R.: Predicative programming, Part I and II. Commun. ACM **27**, 134–151 (1984)
22. Hehner, E.C.R., Hoare, C.A.R.: A more complete model of communicating processes. TCS **26**, 105–120 (1983)
23. Hennessy, M.: Acceptance trees. J. Assoc. Comput. Mach. **32**, 896–928 (1985)
24. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. J. Assoc. Comput. Mach. **32**, 137–161 (1985)

25. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM **21**, 666–677 (1978)
26. Hoare, C.A.R.: A model for communicating sequential processes. In: On the Construction of Programs (R.M. McKeag, A.M. McNaghton, eds.), pp. 229–243. Cambridge: University Press 1980
27. Hoare, C.A.R.: A calculus of total correctness for communicating processes. Sci. Comput. Program. **1**, 49–72 (1981)
28. Hoare, C.A.R.: Specifications, programs and implementations. Tech. Monograph PRG-29, Progr. Research Group. Oxford University 1982
29. Hoare, C.A.R.: Communicating sequential processes. London: Prentice Hall 1985
30. INMOS: OCCAM programming manual. London: Prentice Hall 1984
31. Jorrand, P.: Specification of communicating processes and process implementation correctness. In: Proc. 5th Intern. Symp. on Programming. (M. Dezani-Ciancaglini, U. Montanari, eds.). Lect. Notes Comput. Sci. **137**, 242–256 (1983)
32. Keller, R.: Formal verification of parallel programs. Commun. ACM **19**, 371–384 (1976)
33. Levin, G.M., Gries, D.: A proof technique for communicating sequential processes. Acta Inf. **15**, 281–302 (1981)
34. Milner, R.: A calculus of communicating systems. Lect. Notes Comput. Sci. **92** (1980)
35. Milner, R.: A modal characterisation of observable machine-behaviour. In: Proc. 6th Coll. Trees in Algebra and Programming (E. Asteriano, C. Böhm, eds.). Lect. Notes Comput. Sci. **112**, 25–34 (1981)
36. Milner, R.: Calculi for synchrony and asynchrony. TCS **25**, 267–310 (1983)
37. Misra, J., Chandy, K.M.: Proofs of networks of processes. IEEE Trans. Software Eng. **7**, 417–426 (1981)
38. Misra, J., Chandy, K.M., Smith, T.: Proving safety and liveness of communicating processes with examples. In: Proc. 1st ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing, pp. 201–208. Ottawa 1982
39. de Nicola, R.: A complete set of axioms for a theory of communicating sequential processes. In: Proc. Intern. Conf. on Foundations of Computation Theory (M. Karpinski, ed.). Lect. Notes Comput. Sci. **158**, 115–126 (1983)
40. de Nicola, R., Hennessy, M.: Testing equivalences for processes. TCS **34**, 83–134 (1984)
41. Olderog, E.-R.: Specification-oriented programming in TCSP. In: Logics and Models of Concurrent Systems (K.R. Apt, ed.), pp. 397–435. Berlin, Heidelberg, New York: Springer 1985
42. Olderog, E.-R., Hoare, C.A.R.: Specification-oriented semantics for communicating processes (preliminary version). In: Proc. 10th Coll. Automata, Languages and Programming (J. Diaz, ed.). Lect. Notes Comput. Sci. **154**, 561–572 (1983)
43. Owicki, S., Lamport, L.: Proving liveness properties of concurrent programs. ACM TOPLAS **4**, 455–495 (1982)
44. Plotkin, G.D.: An operational semantics for CSP. In: Formal Description of Programming Concepts II (D. Bjørner, ed.), pp. 199–223. Amsterdam: North Holland 1983
45. Reinecke, R.: Networks of communicating processes: a functional implementation. Manuscript, Dept. Comput. Sci., Univ. of Kaiserslautern 1983
46. Roscoe, A.W.: A mathematical theory of communicating processes. D. Phil., Thesis, Oxford Univ. 1982
47. Rounds, W.C., Brookes, S.D.: Possible futures, acceptances, refusals and communicating processes. In: Proc. 22nd IEEE Symp. on Foundations of Computer Science, Nashville, Tennessee 1981
48. Smyth, M.B.: Power domains. J. Comput. Syst. Sci. **16**, 23–26 (1978)
49. Soundararajan, N., Dahl, O.-J.: Partial correctness semantics of communicating sequential processes. Research Rep. No. 66, Inst. of Informatics, Univ. of Oslo 1982
50. Stoy, J.E.: Denotational semantics: the Scott-Strachey approach to programming language theory. Cambridge: MIT Press 1977
51. Winskel, G.: Events in computation. Ph. D. Thesis, Dept. Comput. Sci., Univ. of Edinburgh 1980
52. Zwiers, J., de Roever, W.P., van Emde Boas, P.: Compositionality and concurrent networks. In: Proc. 12th Coll. Automata, Languages and Programming (W. Brauer, ed.). Lect. Notes Comput. Sci. **194**, 509–519 (1985)

Received February 28, 1984 / October 22, 1985