SPECIFICATION-ORIENTED SEMANTICS FOR COMMUNICATING PROCESSES

by

E.-R. Olderog

and

C.A.R. Hoare

Technical Monograph PRG-37

February 1984

Oxford University Computing Laboratory,
Programming Research Group,
8-11 Keble Road,
Oxford OX1 3QD

[1] Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, 2300 Kiel 1, Fed. Rep. Germany

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

A preliminary version of this paper appeared in [35].

Summary

A process P satisfies a specification S, abbreviated P sat S, if every observation we can make about 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 notion of correctness expressed by P sat S. Safety properties are treated by all models; the more sophisticated models also permit proofs of 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 consistency of the denotational models w.r.t. a simple operational semantics based on transitions which make internal process activity explicit.

Contents

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. [4, 22, 27]). 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 [4, 5, 8, 9, 29, 30, 33]. Let us now outline the approach of our paper.

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 [18]. But the emphasis is here on studying the fundamental concepts involved rather than presenting a full programming notation such as [34]. 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 (Section 3).

(1) Correctness. A process P satisfies a specification S, abbreviated P sat S, if every observation we can make about 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 set Obs of observations together with a simple algebraic structure and define specifications S as certain subsets of Obs reflecting this structure (Section 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 [41] : $S_1 \sqsubseteq S_2$ means that S_2 is more deterministic than S_1.

A specification-oriented semantics assigns denotationally to every process P a specification $\mathcal{M}[\![P]\!]$ such that P sat S is expressed by $S \sqsubseteq \mathcal{M}[\![P]\!]$, i.e. $\mathcal{M}[\![P]\!]$ is within the range of nondeterminism permitted by S. (Section 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 $\mathcal{M}[\![\cdot]\!]$ 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 (Section 6).

Secondly, observations themselves are disallowed 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 (Section 13).

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

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

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 the crucial hiding operators do not. 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

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

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 (Sections 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 $\mathcal{C}$ which reflects the idea of separate channel histories [20, 21]. We show that $\mathcal{C}$ can deal adequately only with acyclic or tree-like networks of processes. Arbitrary networks require the Trace Model $\mathcal{T}$ instead [19, 30]. However, both $\mathcal{C}$ and $\mathcal{T}$ are unable to deal satisfactory with diverging processes. This requires a further refinement of our observations leading to the Divergence Model $\mathcal{D}$.

In $\mathcal{C}$, $\mathcal{T}$ and $\mathcal{D}$ only safety properties can be described by P sat S. Also the concept of external nondeterminism is not yet available. Dea-

ling with the full language of Communicating Processes and with simple liveness properties calls for the more sophisticated Readiness Model $\mathcal{R}$ [12, 20]. We characterise the kind of liveness properties which are expressible in $\mathcal{R}$. By studying the algebraic laws of $\mathcal{R}$ we find that there are finite, i.e. non-recursive processes which are not reducible to simple nondeterministic ones. This shortcomming of $\mathcal{R}$ is avoided in the Refusal or Failure Model $\mathcal{F}$ [22, 7, 39] which makes slightly more identifications among processes than $\mathcal{R}$ - without affecting the expressibility of its liveness properties. In all these models a continuous hiding operator is available in full generality. The relationships 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 (Sections 12 and 13). We show that the models $\mathcal{D}$, $\mathcal{R}$ and $\mathcal{F}$ are fully consistent with the operational semantics, whereas $\mathcal{C}$ and $\mathcal{T}$ 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 (Section 14).

2. Preliminaries

This section describes the general format of our programming language and denotational semantics (cf. [13,14,43]).

A signature Σ consists of a set ($\xi \in$) $Id(\Sigma)$ of identifiers and a set ($f \in$) $Op(\Sigma)$ of operator symbols each one with a certain arity $n \geq 0$. Every signature Σ determines a simple programming language, namely the set ($P,Q \in$) $Rec(\Sigma)$ of (recursive) terms over Σ as defined by the following BNF-like syntax:

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

The recursive construct $\mu\xi._{-}$ [3] defines a binding occurrence of ξ and induces the usual notions of free and bound identifiers. By $CRec(\Sigma)$ we denote the set of all closed recursive terms, i.e. without free identifiers. $FRec(\Sigma)$ denotes the set of all finite, i.e. closed terms $P \in CRec(\Sigma)$ without any occurrence of a recursive construct $\mu\xi._{-}$ inside P.

For $\mu\xi.P$, $Q \in CRec(\Sigma)$ 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 $\mu\xi.P$, 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 $\mathcal{D}$ be a partial order w.r.t. $\sqsubseteq$. A subset $X \subseteq \mathcal{D}$ is directed if every finite subset of X has an upper bound in X. $\mathcal{D}$ is a cpo (complete partial order) if it has a least element $\bot$ and if every directed subset $X \subseteq \mathcal{D}$ has a limit (least upper bound) $\bigsqcup X$ in $\mathcal{D}$. If $\mathcal{D}$ is a cpo, so is $\mathcal{D}^n = \mathcal{D} \times \ldots \times \mathcal{D}$ (n times), with componentwise ordering.

An operator $\Phi : \mathcal{D} \longrightarrow \mathcal{E}$ from one cpo $\mathcal{D}$ into another cpo $\mathcal{E}$ 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 $X \subseteq \mathcal{D}$. Of course continuity implies monotonicity. We remark that an n-place operator $\Phi : \mathcal{D}^n \longrightarrow \mathcal{D}$ is continuous iff it is continuous in every place.

By Knaster-Tarski's fixed point theorem every monotonic operator $\Phi : \mathcal{D} \longrightarrow \mathcal{D}$ has a least fixed point $fix\,\Phi$ in $\mathcal{D}$. If Φ is also

continuous, $\underline{\text{fix}}\ \Phi$ can be represented as

$$\underline{\text{fix}}\ \Phi = \bigsqcup \{ \Phi^n(\bot) \mid n \geq 0 \}$$

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

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

$$\mathcal{M}[\![\cdot]\!] : Rec(\Sigma) \longrightarrow (Val \longrightarrow \mathcal{D}_{\mathcal{M}})$$

is given by

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

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

(iii) $\mathcal{M}[\![\mu \xi . P]\!](\mathcal{V}) = \underline{\text{fix}}\ (\lambda d.\ \mathcal{M}[\![P]\!](\mathcal{V}[d/\xi]))$

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

In the following we assume that there exists some 0-ary symbol $f \in Op(\Sigma)$ with $f_{\mathcal{M}} = \bot$. For simplicity let $\bot$ itself denote this symbol. For $P \in CRec(\Sigma)$ let $P_\bot$ be that finite term which results from P by replacing every occurrence of the form $\mu \xi . R$ in P by $\bot$. For $P, Q \in CRec(\Sigma)$ we write $P \vdash Q$ if Q results from P by replacing one occurrence of the form $\mu \xi . R$ by $R(\mu \xi . R)$. Terms $Q_\bot$ with $P \vdash^* Q$ (reflexive, transitive closure) are called finite (syntactic) approximations of P. Note that $P \vdash^* Q$ implies $\mathcal{M}[\![P]\!] = \mathcal{M}[\![Q]\!]$ and $\mathcal{M}[\![Q_\bot]\!] \sqsubseteq \mathcal{M}[\![P]\!]$. The family $\{ \mathcal{M}[\![Q_\bot]\!] \mid P \vdash^* Q \}$ is directed, and the continuity of the operators in $\mathcal{M}$ implies

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

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 $P,Q \in CRec(\Sigma)$ such that $\mathcal{M}[\![P]\!] = \mathcal{M}[\![Q]\!]$ holds. For signatures $\Sigma 1$ and $\Sigma 2$ we write $\Sigma 1 \subseteq \Sigma 2$ if $Op(\Sigma 1) \subseteq Op(\Sigma 2)$ holds. Let $\mathcal{M}$ be a model for $CRec(\Sigma 2)$ and $\Sigma 1 \subseteq \Sigma 2$. Then the $\Sigma 1$-reduct $\mathcal{M} \restriction \Sigma 1$ is that model for $CRec(\Sigma 1)$ which consists of the cpo $\mathcal{D}_{\mathcal{M}}$ of $\mathcal{M}$ and the subset $\{ f_{\mathcal{M}} \mid f \in Op(\Sigma 1) \}$ of operators.

Let $\mathcal{M}, \mathcal{N}$ be models for $CRec(\Sigma)$. A (weak) homomorphism from $\mathcal{M}$ to $\mathcal{N}$ is an operator $\Phi: \mathcal{D}_{\mathcal{M}} \longrightarrow \mathcal{D}_{\mathcal{N}}$ 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 $f \in Op(\Sigma)$ and $d_1,\ldots,d_n \in \mathcal{D}_{\mathcal{M}}$.

Proposition 2.1 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 $P \in CRec(\Sigma)$.

Proof. As above we may assume $\bot \in Op(\Sigma)$. Continuity of Φ leads to

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

for every $P \in CRec(\Sigma)$ due to (*). Strictness and the (weak) homomorphism property of Φ yields

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

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

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

$$\mathcal{A} \otimes \mathcal{B} = \{ A \times B \mid A \in \mathcal{A} \text{ and } B \in \mathcal{B} \}.$$

Note that $\mathcal{A} \otimes \mathcal{B} \neq \mathcal{A} \times \mathcal{B}$. For relations $g \subseteq A \times B$ and subsets $X \subseteq A$, $Y \subseteq B$ let $g(X) = \{ b \mid \exists a \in X: a\ g\ b \}$ and $g^{-1}(Y) = \{ a \mid \exists b \in Y: a\ g\ b \}$.

For singleton sets we write $g(x)$ and $g^{-1}(y)$ instead of $g(\{x\})$ and $g^{-1}(\{y\})$. We call $g \subseteq A \times B$ domain 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

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

A relation $\longrightarrow \subseteq A \times A$ is called well-founded if there is no infinite chain

$$\ldots \longrightarrow a_3 \longrightarrow a_2 \longrightarrow a_1$$

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

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

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

where $\overline{\ }$ 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 $CRec(\Sigma)$ 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 [18] or OCCAM [34].

Formally, we start from a finite set or alphabet $(a,b \in)$ Comm of communications. In OCCAM e.g. Comm is structured as Comm = Cha × $\mathcal{M}$ 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 Σ of Communicating Processes consists of a set $Id(\Sigma)$ of identifiers ξ and the set

$$Op(\Sigma) = \{ \underline{stop}, \underline{div} \} \cup \{ a \rightarrow \mid a \in Comm \} \cup \{ \underline{or}, \Box \} \cup \{ \|_A \mid A \subseteq Comm \} \cup \{ \setminus b \mid b \in Comm \}$$

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

$$P ::= \underline{stop} \mid \underline{div} \mid a \rightarrow P \mid P \ \underline{or} \ Q \mid P \ \Box \ Q \mid P \|_A Q \mid P \setminus b \mid \xi \mid \mu \xi . P$$

The closed terms in $CRec(\Sigma)$ 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 [11]: 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 [11]: 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 [18] P or Q reflects the concept of a guarded command with true guards [true $\longrightarrow$ P $\Box$ true $\longrightarrow$ 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 ($\|_\emptyset$) to full synchrony ($\|_{Comm}$). 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 non-interleaving semantics (cf. e.g. [29]).

P \ 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 omitted full sequential composition P;Q. There is, however, no difficulty in modelling this concept [19,22]. Also - if Q does not diverge - the effect of P;Q can be simulated by parallel composition plus hiding, i.e. we can define

$$P;Q = (P \|_{\{\surd\}} (\surd \longrightarrow Q)) \setminus \surd$$

where the special communication $\surd$ reports successful termination of P [19].

Besides the full language CRec(Σ) we consider two sublanguages CRec($\Sigma 1$) and CRec($\Sigma 2$) with $\Sigma 1 \subseteq \Sigma 2 \subseteq \Sigma$.

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

4. Observations and Specifications

It is quite easy to express the intuitive understanding of processes operationally in terms of transitions (see Section 12). But this formalisation has one severe disadvantage: it does not abstract from internal actions. Such an abstraction is essential if we want to compose large processes hierarchically as networks of smaller ones and prove that these networks meet a given specification [30].

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, written as

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

if every observation we can make about 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 about 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 $\longrightarrow$ between observations which reflects their possible ordering in time: $x \longrightarrow y$ means that observation y can be made immediately after x, without intervening observation.

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

(O1) $\longrightarrow$ is well-founded.

(O2) $\longrightarrow$ is domain and image finite.

Condition (O2) simplifies the development of the theory (cf. Definitions 4.3 and 8.2). Let

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

Note that Min $\neq \emptyset$ because of condition (O1) and Obs $\neq \emptyset$. By a grounded chain of length $n \geq 0$ for x we mean a chain $x_0 \longrightarrow \ldots \longrightarrow x_n = x$ with $x_0 \in$ Min. Due to (O1) we can assign a level $\| x \|$ to every observation x:

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

Informally $\| x \|$ measures the minimal progress a process has made up to observation x. Note that $x \longrightarrow y$ implies $\| y \| \leq \| x \| + 1$.

Example 4.2 Consider as observations the set

$$(s,t \in) \quad \text{Obs}_{\mathcal{T}} = \text{Comm}^*$$

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

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

Then $(\text{Obs}_{\mathcal{T}}, \longrightarrow)$ is a simple observation space. And $s \stackrel{*}{\longrightarrow} t$ holds iff s is a prefix of t, abbreviated $s \leq t$. Note that here every trace s has exactly one grounded chain

$$\varepsilon \longrightarrow \ldots \longrightarrow s .$$

The assumption of a finite alphabet Comm (see Section 3) allows us to work with image finite relations throughout this paper, a substantial simplification of the theory. //

Given a simple observation space $(\text{Obs}, \longrightarrow)$ we can now talk about specifications over $(\text{Obs}, \longrightarrow)$: these are by definition simply sets $S \subseteq \text{Obs}$ of observations. 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 [41]:

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

$S_1 \sqsubseteq S_2$ 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 $\mathcal{M}$ for $CRec(\Sigma)$ which assign to every process $P \in CRec(\Sigma)$ a specification $\mathcal{M}[\![P]\!]$ using ordering $\subseteq$. But to do so the $\mathcal{M}[\![P]\!]$ must be special sets of observations, called here process specifications, which reflect the algebraic structure of Obs.

Definition 4.3 A process specification over $(Obs, \longrightarrow)$ is a subset $S \subseteq Obs$ with:

(S1) S includes Min: $Min \subseteq S$.

(S2) S is generable: $\forall x \in S - Min \ \exists y \in S: y \longrightarrow x$.

A specification space over $(Obs, \longrightarrow)$ is a family $(S, T \in)$ Spec $\subseteq \mathcal{P}(Obs)$ of process specifications such that Spec forms a cpo w.r.t. ordering $\supseteq$.

Since $\longrightarrow$ is domain finite, the set of all process specifications over $(Obs, \longrightarrow)$ forms a specification space, called the full specification space.

Example 4.4 Take $(Obs_{\mathcal{T}}, \longrightarrow)$ of Example 4.2. A subset $S \subseteq Comm^*$ is called prefix-closed if $t \in S$ and $s \leq t$ always imply $s \in S$. Then the set $Spec_{\mathcal{T}}$ of all prefix-closed subsets $S \subseteq Comm^*$ with $\varepsilon \in S$ is the full specification space over $(Obs_{\mathcal{T}}, \longrightarrow)$ [19]. //

If $Spec_1$ and $Spec_2$ are specification spaces over $(Obs_1, \longrightarrow_1)$ and $(Obs_2, \longrightarrow_2)$ then the cartesian product $Spec_1 \times Spec_2$ is of course a cpo with componentwise ordering $\supseteq$ (cf. Section 2). Additionally, the sets $S \times T \in Spec_1 \otimes Spec_2$ (inner product) are process specifications over the product observation space

$$(Obs_1 \times Obs_2, \longrightarrow_{12})$$

where $\longrightarrow_{12}$ is the following "interleaving relation" on $Obs_1 \times Obs_2$:

$$(x_1, x_2) \longrightarrow_{12} (y_1, y_2) \text{ if either } x_1 \longrightarrow_1 y_1 \text{ and } x_2 = y_2$$
$$\text{or } x_1 = y_1 \text{ and } x_2 \longrightarrow_2 y_2$$

i.e. the pair makes a step when either component makes a step.

Since the natural ordering $\geq$ on $Spec_1 \otimes Spec_2$ is isomorphic with the componentwise ordering on $Spec_1$ and $Spec_2$ in the sense that

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

holds for all $S_1, T_1 \in Spec_1$ and $S_2, T_2 \in Spec_2$, it follows that the inner product $Spec_1 \otimes Spec_2$ is indeed a specification space over $(Obs_1 \times Obs_2, \multimap_{12})$.

5. Specification-oriented Semantics

We can now be precise about the desired form of semantics.

Definition 5.1 Let $\Sigma O \subseteq \Sigma$. A specification-oriented model for $CRec(\Sigma O)$ over $(Obs, \longrightarrow\!\!\triangleright)$ consists of a specification space Spec over $(Obs, \longrightarrow\!\!\triangleright)$ and a set $\{ f_{\mathcal{M}} \mid f \in Op(\Sigma O) \}$ of $\supseteq$-continuous operators on process specifications $S \in Spec$.
A specification-oriented semantics for $CRec(\Sigma O)$ is a denotational semantics $\mathcal{M}[\![\cdot]\!]$ induced by a specification-oriented model $\mathcal{M}$ for $CRec(\Sigma O)$.

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

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

This is of course an invariance principle: every observation assigned to P by $\mathcal{M}$ should be allowed by S. However, as we shall see later on, by varying the structure of observations both safety and (certain) liveness properties of Communicating Processes can be expressed in terms of sat.

This clarifies the domains of our sematical models $\mathcal{M}$ and the notion of process correctness. Next we exploit the simple algebraic structure of observations and process specifications to derive some results for constructing $\supseteq$-continuous operators on specifications.

Let $Spec_1$ and $Spec_2$ always denote specification spaces over $(Obs_1, \longrightarrow\!\!\triangleright_1)$ and $(Obs_2, \longrightarrow\!\!\triangleright_2)$. We wish to construct $\supseteq$-continuous operators

$$C_g : Spec_1 \longrightarrow \mathcal{P}(Obs_2)$$

working on process specifications by starting from relations

$$g \subseteq Obs_1 \times Obs_2$$

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

The simplest way of achieving this is to take the pointwise extension $\mathcal{O}_g$: $Spec_1 \longrightarrow \mathcal{P}(Obs_2)$ defined by

$$\mathcal{O}_g(S) = \{ y \in Obs_2 \mid \exists x \in S: x\ g\ y \} = g(S) .$$

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 domain 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 $S_i \in Spec_1$

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

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 domain finite relations g. But the important hiding operators are not.

Example 5.1 Take $Spec_{\mathcal{T}}$ of Example 4.4. For observations (i.e. traces) $s,t \in Comm^*$ we define

$$s\ g\ t \text{ iff } s \setminus b = t$$

where $s \setminus b$ is obtained from s by deleting or hiding all occurrences of b in s. Then g is not domain finite; and indeed $\mathcal{O}_g$: $Spec_{\mathcal{T}} \longrightarrow \mathcal{P}(Comm^*)$ is not $\supseteq$-continuous as soon as $|Comm| \geq 2$ holds. Take e.g.

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

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

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

//

We present now an abstract analysis of such hiding operators based on the relation $\longrightarrow$ between observations. First we introduce a new operator $\mathcal{O}_g^{\infty} : \mathrm{Spec}_1 \longrightarrow \mathcal{P}(\mathrm{Obs}_2)$ 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 $C_g : \mathrm{Spec}_1 \longrightarrow \mathcal{P}(\mathrm{Obs}_2)$ 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 $l \geq 0$ 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.

Note: if the set $\mathrm{Min}_1 \subseteq \mathrm{Obs}_1$ of "minimal" observations in Obs_1 is finite, every relation $g \subseteq \mathrm{Obs}_1 \times \mathrm{Obs}_2$ is level finite. This will be the case in all our applications to Communicating Processes.

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 $\{ S_i \mid i \in I \}$ 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 $Y = \{ y \mid y \xrightarrow{*}_2 y' \}$. Then Y is finite because $\xrightarrow{*}_2$ is well-founded and domain finite. Thus by the definition of C_g:

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

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

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

(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

$$x_o \longrightarrow_1 \cdots \longrightarrow_1 x_m = x$$

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

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

holds. Commutativity of g implies

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

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 $\| x_i \| \leq l$ by the definition of l.

Contradiction.

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

We remark that the proof of Theorem 5.5 does not use the general assumption that the relations $\longrightarrow_1$ and $\longrightarrow_2$ are image finite.

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 $S_o \supseteq \ldots \supseteq S_n \supseteq \ldots$ we get

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

//

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 Section 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 $\mathcal{C}$

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 $CRec(\Sigma 1)$ of Communicating Processes where parallel composition is restricted to the case of $|A| \leq 1$. We imagine that the only thing we can observe about a process P is how many times each communication $a \in Comm$ has occurred up to a given moment [21]. Formally, we define the set of observations by

$$(h \in)\ Obs_{\mathcal{C}} = Comm \longrightarrow \mathbb{N}_0$$

i.e. for each communication a there is a separate counter. $Obs_{\mathcal{C}}$ is a simple observation space with the following relation $\longrightarrow$:

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

i.e. h' differs from h in that exactly one counter is incremented by 1. Then $h \xrightarrow{*} h'$ means that $h(a) \leq h'(a)$ holds for every $a \in Comm$ ($h \leq h'$ for short). Let ZERO denote the constant mapping h with $h(a) = 0$ for every $a \in Comm$. As process specifications we take the full specification space $Spec_{\mathcal{C}}$ consisting of all generable subsets $S \subseteq Obs_{\mathcal{C}}$ with $ZERO \in S$.

The Counter Model $\mathcal{C}$ is now given by $Spec_{\mathcal{C}}$ and the following set $\{ f_{\mathcal{C}} \mid f \in Op(\Sigma 1) \}$ of operators on process specifications S which formalise the intuitions about processes explained in Section 3 (since $\mathcal{C}$ remains constant throughout this section, we drop all indices $\mathcal{C}$ at operators $f_{\mathcal{C}}$):

(1) stop = $\{ ZERO \}$

(2) div = $Obs_{\mathcal{C}}$

This definition reveals a general strategy of specification-oriented semantics, namely to identify all "undesirable parts"

of processes P with S = Obs, the weakest specification in the $\supseteq$-ordering. By a "part" of P we mean the subsequent behaviour of P after some initial observations have been made. In this paper we will consider as "undesirable" all parts of processes which can diverge right from the beginning, i.e. engage in an infinite sequence of internal (hidden) actions. The simplest example of such a part is the process div itself. In general a divergence can be introduced into a process either explicitly via div or explicitly via recursion or hiding (see below). These ideas will be discussed again in Section 8 and made precise later in Sections 12 and 13.

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

To ensure that this operator is $\supseteq$-continuous we check the relation g with

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

Clearly $a \longrightarrow S = \mathcal{O}_g(S)$. Since g is domain finite, Proposition 5.2 implies the $\supseteq$-continuity of $\mathcal{O}_g$. Also it is easy to see that $\mathcal{O}_g$ preserves the generability of S.

(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. Section 4).

(5) $S_1 \parallel_A S_2 = \{h \mid \exists h_1 \in S_1 \; \exists h_2 \in S_2 : (h_1,h_2) \; g \; h\} = \mathcal{O}_g(S_1,S_2)$

where g relates the product $Obs_\varphi \times Obs_\varphi$ with Obs_φ by:

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

This formalises the intuition that S_1 and S_2 work independently except for communications mentioned in A. Clearly g is domain finite, and thus $\mathcal{O}_g$ $\supseteq$-continuous by Proposition 5.2. But $\mathcal{O}_g$ preserverves generability of specifications S_1 and S_2 only thanks to the restriction $|A| \leq 1$ in $\Sigma 1$.

For $|A| \geq 2$ our simple definition of $\|_A$ does not necessarily ensure generability. For example, $S_1 = a \longrightarrow b \longrightarrow \underline{stop}$ and $S_2 = b \longrightarrow a \longrightarrow \underline{stop}$ denote generable specifications in $Spec_{\mathcal{C}}$, but

$$S_1 \|_{\{a,b\}} S_2 = \{ZERO\} \cup \left\{ h \;\middle|\; \begin{array}{l} h(a) = h(b) = 1 \\ \wedge\ \forall c \neq a,b\colon h(c) = 0 \end{array} \right\}$$

does not. Informally this is because we cannot observe relative timing between different communications in the Counter Model $\mathcal{C}$. A similar problem, known as the merge anomaly, can arise in loosely coupled nondeterministic dataflow networks [6,9]. As we shall see now, generability of process specifications is vital for proving the continuty of the hiding operator $\backslash b$.

(6) $S \backslash b$: we consider the relation $g \subseteq Obs_{\mathcal{C}} \times Obs_{\mathcal{C}}$ with:

(*) $h\ g\ h'$ iff $h'(b) = 0$ and $\forall a \neq b\colon h(a) = h'(a)$.

Intuitively, g hides all communications b in h. Note that g is not domain finite any more. And indeed, $\mathcal{O}_g$ is not $\supseteq$-continuous as can be shown analogously to Example 5.3. 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:

$$S \backslash b = \{ h \mid h(b) = 0 \wedge \exists n \geq 0\colon h[n/b] \in S \} \cup \{ h' \mid \exists h \leq h'\ \overset{\infty}{\exists} n \geq 0\colon h[n/b] \in S \}$$

The infinity clause of this definition accords with the principle of specification-oriented semantics to identify "undesirable parts" of $S \backslash b$ where infinitely many hidden b's are possible with the full set $Obs_{\mathcal{C}}$. We remark that $S \backslash b$

preserves the generability of $S \in Spec_{\mathcal{C}}$.

It is interesting to note that $S \setminus b = \mathcal{O}_g \cup \mathcal{O}_g^{\infty}$ is not $\subseteq$-continuous for arbitrary specifications $S \subseteq Obs_{\mathcal{C}}$. Look e.g. at

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

for $n \geq 0$. Of course

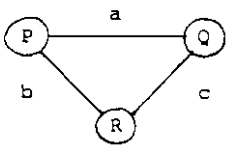
$$S_0 \supseteq \ldots \supseteq S_n \supseteq \ldots$$

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

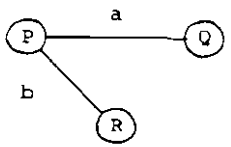
$$\bigcap_n (S_n \setminus b)^{\cdot} = Obs_{\mathcal{C}} \neq \{ZERO\} = (\bigcap_n S_n) \setminus b .$$

Thus <u>generability</u> of specifications is essential for the continuity of the above hiding operator $S \setminus b$.

This finishes the definition of the Counter Model $\mathcal{C}$. It induces a specification-oriented semantics $\mathcal{C}[\![\cdot]\!]$ for $CRec(\Sigma 1)$ according to Sections 3 and 5. 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 $|A| \leq 1$ in $\Sigma 1$ means that we can construct only <u>acyclic</u> or <u>tree-like</u> networks. For example, we cannot construct the cyclic network



We can at best construct a tree-like subnet of it, e.g. by $(P \parallel_{\{a\}} Q) \parallel_{\{b\}} R$ we get

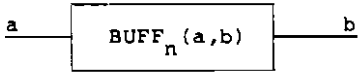


with arc c missing between Q and R. Next we will study a typical example for tree-like networks: a chain of buffers.

Example 6.1 Consider for $n \geq 1$ and $a,b \in Comm$ the following specification:

$$BUFF_n(a,b) = \{\, h \mid h(b) \leq h(a) \leq h(b)+n \wedge \forall d \neq a,b: h(d) = 0 \,\}$$

Then $BUFF_n(a,b)$ 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 $BUFF_n(a,b) \in Spec_{\mathcal{C}}$.

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

$$P(a,b) = \mu\xi.(a \rightarrow b \rightarrow \xi)$$

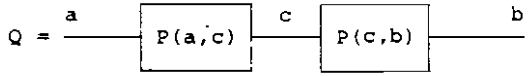
we get

$$\mathcal{C}[\![\, P(a,b) \,]\!] = BUFF_1(a,b) \;.$$

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 built $BUFF_2(a,b)$ we first construct a "chain" Q of two 1-place buffers:

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

or in graphical terms



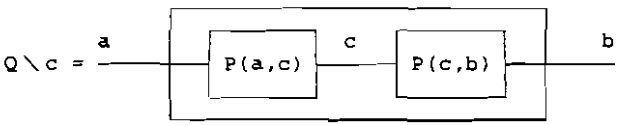
The resulting process behaves like a 2-place buffer except for the intermediate communications c:

$$\mathcal{C}[\![\, Q \,]\!] = \left\{ h \;\middle|\; \begin{array}{l} h(b) \leq h(c) \leq h(a) \leq h(c)+1 \leq h(b)+2 \\ \wedge\ \forall d \neq a,b,c:\ h(d) = 0 \end{array} \right\}$$

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

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

or in graphical terms



This construction yields indeed

$$\mathcal{C}[\![\, (P(a,c) \parallel_{\{c\}} P(c,b)) \setminus c \,]\!] = BUFF_2(a,b) \ .$$

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

$$R = a \longrightarrow \mu\xi.((a \rightarrow b \longrightarrow \xi) \ \underline{or}\ (b \rightarrow a \longrightarrow \xi))$$

with $\mathcal{C}[\![R]\!] = BUFF_2(a,b)$. //

In the example we dealt with semantic <u>equality</u>, 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 induced by <u>sat</u>. Clearly

$$\mathcal{C} \vdash P(a,b) \ \underline{sat}\ BUFF_1(a,b) \text{ and}$$

$$\mathcal{C} \vdash Q \setminus c \ \underline{sat}\ BUFF_2(a,b) \ .$$

But since $BUFF_n(a,b) \subseteq BUFF_{n+1}(a,b)$, we also have

$$\mathcal{C} \models P(a,b) \ \underline{sat}\ BUFF_{n+1}(a,b) \text{ and}$$

$$\mathcal{C} \models Q \setminus c \ \underline{sat}\ BUFF_{n+2}(a,b)$$

for all $n \geq 0$. This means that we cannot use correctness reasoning based on $\mathcal{C} \vdash P(a,b) \underline{sat}\ BUFF_n(a,b)$ to ensure that a buffer has a capacity of <u>at least</u> n.

What is worse, since

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

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; this will be treated fully in Sections 9 et seq. First let us refine the Counter Model $\mathcal{C}$ to a model which can deal with cyclic networks.

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

To treat CRec(Σ 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

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

of traces over Comm [19,30]. $\mathrm{Obs}_{\mathcal{T}}$ induces the simple observation space $(\mathrm{Obs}_{\mathcal{T}}, \longrightarrow\!\!\triangleright\ )$ and the full specification space $\mathrm{Spec}_{\mathcal{T}}$ explained in Examples 4.2 and 4.4.

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

(1) $\underline{\mathrm{stop}} = \{\varepsilon\}$

(2) $\underline{\mathrm{div}} = \mathrm{Obs}_{\mathcal{T}}$

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

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

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

Here $t_1 \parallel_A t_2$ 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 $t_1 \parallel_A t_2$ inductively as follows:

(i) $\varepsilon \parallel_A \varepsilon = \{\varepsilon\}$

(ii) $as \parallel_A \varepsilon = \varepsilon \parallel_A as =$

$$\begin{cases} \emptyset & \text{if } a \in A \\ a \cdot (s \parallel_A \varepsilon) & \text{if } a \notin A \end{cases}$$

(iii) $as \parallel_A bt = bt \parallel_A as =$

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

$$(6)\quad S \setminus b = \{\, s \setminus b \mid s \in S \,\} \cup \{\, (s \setminus b)t \mid t \in Comm^* \wedge \forall n \geq 0\colon sb^n \in S \,\}$$

where $s \setminus b$ results from s by removing all occurrences of b in s.

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

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

$$s\ g\ h \text{ iff } \forall a \in A\colon h(a) = a\#s$$

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

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

Proof. By Proposition 2.1 it suffices to show that $\mathcal{O}_g$ is a strict and $\supseteq$-continuous homomorphism from the reduct $\mathcal{T} \restriction \Sigma 1$ to $\mathcal{C}$. Since g is domain finite, the $\supseteq$-continuity of $\mathcal{O}_g$ follows from Proposition 5.2. The homomorphism property of $\mathcal{O}_g$ (which implies here strictness) is easy to verify; only the parallel composition needs some care:

$$\mathcal{O}_g(S_1 \parallel_{A\,\mathcal{T}} S_2) = \mathcal{O}_g(S_1) \parallel_{A\,\mathcal{C}} \mathcal{O}_g(S_2)$$

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

If we assume a channel structure $Comm = Cha \times \mathcal{M}$ of communications, an interesting combination $\mathcal{C}\&\mathcal{T}$ of the two models $\mathcal{C}$ and $\mathcal{T}$ is possible, viz. when we postulate that the relative order between

communications can be observed if and only if they are sent along the same channel. We then talk of channel histories. $\mathcal{C}\&\mathcal{T}$ is able to describe networks of processes acyclically connected via channels. Possible applications for $\mathcal{C}\&\mathcal{T}$ are buffers and protocols as demonstrated in [10].

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

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

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

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

$$\underline{div} \parallel_{Comm} P = P$$

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

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

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

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

$$\underline{div} = \mu\xi.((a \rightarrow \xi) \ \underline{or} \ (b \rightarrow \xi))$$

holds in $\mathcal{T}$. This identification explains the unrealistic law $\underline{div} \parallel_{Comm} P = P$ in $\mathcal{T}$.

In this section we improve the Trace Model $\mathcal{T}$ into a Divergence Model $\mathcal{D}$ where the law (*) is valid without qualification. The idea is to separate arbitrary observable nondeterminism without internal divergence from the wholly unpredictable behaviour which includes the possibility of divergence. In $\mathcal{D}$ only the wholly unpredictable behaviour will be modelled by the weakest specification $S = Obs_{\mathcal{D}}$ (see also Definition 8.3).

To realise this idea we first extend the set of observations to

$$Obs_{\mathcal{D}} = Comm^* \cup \{s\uparrow \mid s \in Comm^*\}$$

where $\uparrow$ is a new symbol $\notin$ Comm which is never used explicitly as a communication in a process, but which can appear in a trace. $\mathcal{D}$ will be constructed in such a way that $\uparrow$ appears only in a trace $s\uparrow$ of a process which can diverge from s onwards. Thus $\uparrow$ may be thought of as an observation of divergence. (Of course, basic incomputability results tell us that we cannot expect to effectively observe a divergence; our reason for introducing $\uparrow$ here is to prove later on its absence in particular processes.)

Let s,t range over $Comm^*$. As $\longrightarrow$ we take the smallest relation over $Obs_{\mathcal{D}}$ such that

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

holds for all $s \in Comm^*$ and $a \in Comm$. Then $(Obs_{\mathcal{D}}, \longrightarrow)$ is a simple observation space.

Our second refinement is more substantial. We don't want to take as specification space the set of all generable specifications $S \subseteq Obs_{\mathcal{D}}$ with $\xi \in Obs_{\mathcal{D}}$. Instead we wish to restrict ourselves to those $S \in Obs_{\mathcal{D}}$ which additionally satisfy:

(**) $\quad s\uparrow \in S$ implies sa, $sa\uparrow \in S$

for every $a \in Comm$, i.e. with $s\uparrow$ also all successors of $s\uparrow$ under $\longrightarrow$ should be present in S. This condition makes explicit the principle of specification-oriented semantics to identify "undesirable", i.e. diverging parts of a process with the full set of all possible successor observations, i.e. the weakest possible

specification (cf. Section 6). Informally, once a process is broken, its behavious is wholly unpredictable; and it remains so even after further observations have been made.

The idea to require also a sort of converse (**) of the generability condition for specifications is not only useful for the Divergence Model $\mathcal{D}$ but also fundamental in the following sections. We therefore incorporate this idea into the general framework of observation and specification spaces, and call it the extensibility condition. The simplest definition of such a condition would be the literal converse of generability:

$$\forall x \in S - Max \;\; \exists y \in S: \; x \longrightarrow y$$

where $Max = \{ x \mid \neg \exists y: \; x \longrightarrow y \}$. 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 need to extend the algebraic structure of observation spaces by a second relation $\longrightarrow\!\!\!\!\!\triangleright$ between single observations x and sets Y of successor observations of x. Informally $\longrightarrow$ and $\longrightarrow\!\!\!\!\!\triangleright$ reflect the amount of information we can retrieve from observations.

Definition 8.1 An observation space is a structure $(Obs, \longrightarrow, \longrightarrow\!\!\!\!\!\triangleright)$ where $(Obs, \longrightarrow)$ is a simple observation space with $\longrightarrow$ satisfying conditions (O1) and (O2) of Definition 4.1, and where $\longrightarrow\!\!\!\!\!\triangleright$ is a relation $\longrightarrow\!\!\!\!\!\triangleright \subseteq Obs \times \mathcal{P}(Obs)$ such that

(O3) $x \longrightarrow\!\!\!\!\!\triangleright Y$ implies $x \longrightarrow y$

for all $y \in Y$, i.e. Y is some subset of possible successors of x under the familiar relation $\longrightarrow$.

Note that $\longrightarrow\!\!\!\!\!\triangleright$ is image finite since $\longrightarrow$ itself is image finite. As notations we introduce:

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

$$x \longrightarrow\!\!\!\!\!\triangleright y \text{ abbreviates } x \longrightarrow\!\!\!\!\!\triangleright \{ y \}$$

Simple observation spaces $(Obs, \longrightarrow)$ will from now on be identified with observation spaces $(Obs, \longrightarrow, \longrightarrow\!\!\!\!\!\triangleright)$ where the relation $\longrightarrow\!\!\!\!\!\triangleright$ is empty and thus $MAX = Obs$. If $\longrightarrow$ and $\longrightarrow\!\!\!\!\!\triangleright$ are understood, we

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 subset $S \subseteq Obs$ with:

(S1) S includes Min: $Min \subseteq S$

(S2) S is generable: $\forall x \in S - Min\ \exists y \in S: y \longrightarrow x$

(S3) S is extensible: $\forall x \in S - MAX\ \exists Y \subseteq S: x \longrightarrow\!\!\!\!\rightarrow Y$

A specification space over Obs is a family $Spec \subseteq \mathcal{P}(Obs)$ of process specifications which forms a cpo under $\supseteq$.

Note the symmetry between (S2) and (S3). Since $\longrightarrow$ is domain finite and $\longrightarrow\!\!\!\!\rightarrow$ is image finite, the set of all process specifications over Obs forms again a specification space: the full specification space. If $\longrightarrow\!\!\!\!\rightarrow$ is empty, Definition 8.2 reduces to Definition 4.3. In particular, every specification space over $(Obs, \longrightarrow, \longrightarrow\!\!\!\!\rightarrow)$ is also a specification space over $(Obs, \longrightarrow)$. Thus our results about continuity in Section 5 remain valid as they rely only on the underlying structure $(Obs, \longrightarrow)$. Analogously to Definition 5.1 we define specification-oriented models $\mathcal{M}$ over $(Obs, \longrightarrow, \longrightarrow\!\!\!\!\rightarrow)$.

Let us now continue with the Divergence Model $\mathcal{D}$. We take $\longrightarrow\!\!\!\!\rightarrow$ to be the smallest relation between $Obs_{\mathcal{D}}$ and $\mathcal{P}(Obs_{\mathcal{D}})$ such that

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

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

(2) div = $Obs_{\mathcal{D}}$

(5) $$S_1 \|_A S_2 = \{ s \mid \exists t_1 \in S_1,\ t_2 \in S_2 :\ s \in t_1 \|_A t_2 \}$$
$$\cup \left\{ su \;\middle|\; \begin{array}{l} u \in Obs_{\mathcal{D}} \wedge \exists t_1 \in S_1,\ t_2 \in S_2 : \\ \quad (s \in t_1 \|_A t_2 \wedge (t_1\uparrow \in S_1 \vee t_2\uparrow \in S_2)) \end{array} \right\}$$

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 $S_1 \|_A S_2 = \mathcal{O}_g(S_1, S_2)$ is domain finite. This guarantees $\supseteq$-continuity by Proposition 5.2.

(6) $$S \setminus b = \{ s \setminus b \mid s \in S \}$$
$$\cup \{ (s \setminus b)u \mid u \in Obs_{\mathcal{D}} \wedge \forall n \geq 0 :\ sb^n \in S \}$$

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

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

(i) $\underline{div}\ \underline{or}\ P = P\ \underline{or}\ \underline{div} = \underline{div}$

(ii) $\underline{div} \|_A P = P \|_A \underline{div} = \underline{div}$

(iii) $\underline{div} \setminus b = \underline{div}$

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

Next we wish to relate the models $\mathcal{D}$ and $\mathcal{T}$. As explained earlier, the reason for achieving the law (ii) in $\mathcal{D}$ is the careful distinction between arbitrary observable nondeterminism without internal divergence and the wholly unpredictable behaviour including divergence. This distinction can be made precise by considering the specification

$$S = Obs_{\mathcal{T}} = Comm^* \subseteq Obs_{\mathcal{D}}$$

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

$$\mathcal{D} \models P \text{ sat } Obs_{\mathcal{T}}$$

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

Definition 8.3 A process $P \in CRec(\Sigma 2)$ is called divergence free if $\mathcal{D} \models P \text{ sat } Obs_{\mathcal{T}}$ holds.

Note that whenever $f(P_1,\ldots,P_n) \in CRec(\Sigma 2)$ is divergence free, all $P_1,\ldots,P_n$ are divergence free as well.

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

Proof. (i) $\mathcal{T}[\![P]\!] \subseteq \mathcal{D}[\![P]\!]$:
The operator $\Phi : Spec_{\mathcal{D}} \longrightarrow Spec_{\mathcal{T}}$ with $\Phi(S) = S \cap Comm^*$ is a strict and continuous weak homomorphism (w.r.t. $\supseteq$) from $\mathcal{D}$ to $\mathcal{T}$. (Weakness is due to $\|_A$.) Thus Proposition 2.1. yields

$$\Phi(\mathcal{D}[\![P]\!]) \supseteq \mathcal{T}[\![P]\!]$$

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

(ii) $\mathcal{D}[\![P]\!] = \mathcal{T}[\![P]\!]$ for divergence free P:
We use finite syntactic approximations of P as defined in Section 2. Note that the general symbol $\perp$ of Section 2 is now div: thus we write P_{div} instead of $P_{\perp}$. First compare the operator definitions in $\mathcal{D}$ and $\mathcal{T}$ to realise that

$$(*) \qquad \mathcal{D}[\![f(P_1,\ldots,P_n)]\!] = f_{\mathcal{T}}(\mathcal{D}[\![P_1]\!],\ldots,\mathcal{D}[\![P_n]\!])$$

holds for all divergence free $f(P_1,\ldots,P_n) \in CRec(\Sigma 2)$.

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

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

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

$$\mathcal{V}_{\mathcal{D}}(\xi_i) = \mathcal{D}[\![\mu\xi_i.R_i]\!]$$

$$\mathcal{V}_{\mathcal{T}}(\xi_i) = \mathcal{T}[\![\underline{\mathrm{div}}]\!]$$

for $i = 1, \ldots, n$. Note that

$$(**) \qquad \mathcal{D}[\![\mu\xi_i.R_i]\!] \subseteq \mathcal{T}[\![\underline{\mathrm{div}}]\!]$$

holds because with P also all $\mu\xi_i.R_i$ are divergence free. Thus we get

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

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

$$\begin{aligned}
&\mathcal{D}[\![P]\!] = \bigcap \{ \mathcal{D}[\![Q]\!] \mid P \overset{*}{\vdash\!\!\!-} Q \} \\
&\subseteq \bigcap \{ \mathcal{T}[\![Q_{\underline{\mathrm{div}}}]\!] \mid P \overset{*}{\vdash\!\!\!-} Q \} = \mathcal{T}[\![P]\!]
\end{aligned}$$

due to Section 2. //

9. Safety and Liveness Properties

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

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

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

$P = \underline{stop}$.

This indicates that (*) expresses only safety properties [36] of P in the sense that P does nothing that is forbidden by S. For (*) this means that we can prove:

(a) absence of disallowed traces

(b) absence of divergence.

(The simpler models $\mathcal{T}$ and $\mathcal{C}$ 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 $\{P\}$ div $\{Q\}$. In $\mathcal{D}$ the process div satisfies of course only the weakest specification $Obs_{\mathcal{D}}$.

Let us now turn to the question of liveness properties. Intuitively, liveness means that a process is under all circumstances, i.e. independently of its internal activity, able to perform a certain predefined task [23]. In the following we propose an abstract framework for discussing this idea.

A simple liveness property is a pair

(T,L)

of specifications $T,L \in Spec_{\mathcal{T}}$ (i.e. non-empty, prefix-closed sets $T,L \subseteq Comm^*$ of traces) with $L \subseteq T$. Informally a process

(**) P satisfies (T,L)

if the following holds:

(1) P is divergence free.

(2) P engages only in traces mentioned in T.

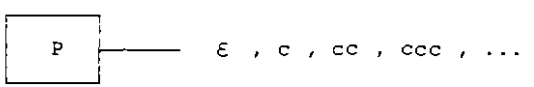
(3) P is able to engage in every trace of L - independently of its internal activity. (Thus L is the "task" mentioned above.)

Conditions (1) and (2) are well-understood from the Divergence Model $\mathcal{D}$. Condition (3) will be explained in the subsequent section by translating every simple liveness property (T,L) into a proper process specification S(T,L) such that (**) is defined by

P 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 $T = L = \{ c^n \mid n \geq 0 \}$. Then

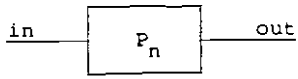
P satisfies (T,L)

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

ε , c , cc , ccc , ...

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 ∈ Comm we use here the suggestive names in,out ∈ Comm for the communications of P_n:



Take

$$T_n = L_n = \{ s \mid s \in \{in,out\}^* \wedge out\#s \leq in\#s \leq (out\#s)+n \}$$

where in#s (out#s) denotes the number of in's (out's) in s (cf. Section 7). Then

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

If according to T_n the process P_n engages only in communications in(put) and out(put) such that the number of outputs never exceeds the number of inputs and the number of inputs never exceeds the number of outputs 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 $in\#s < (out\#s)+n$, it should accept another input.

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

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

We generalise this concept of a simple liveness property as follows: a (general) liveness property is a pair

$$(T, \mathcal{L})$$

with $T \in \text{Spec}_\mathcal{T}$ and a non-empty $\mathcal{L} \subseteq \text{Spec}_\mathcal{T}$ such that $L \subseteq T$ holds for every $L \in \mathcal{L}$. (Simple liveness properties are identified with pairs $(T, \mathcal{L})$ where $|\mathcal{L}| = 1$.) We define:

$$P \text{ satisfies } (T, \mathcal{L}) \text{ if } \exists L \in \mathcal{L} : P \text{ satisfies } (T, L).$$

Intuitively, P satisfies $(T, \mathcal{L})$ if P is able to engage in every

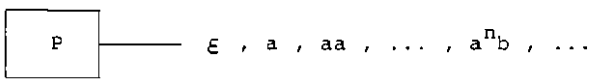
trace of at least one $L \in \mathcal{L}$. Thus a simple liveness property fixes one particular process behaviour whereas a general liveness property describes only a general pattern of a desired behaviour.

Example 9.3 Surprisingly, we can view the concept of deadlock freedom (which is often classified as a safety property [36]) as a general liveness property $(T, \mathcal{L})$ with

$$T = Comm^* \text{ and } \mathcal{L} = \{ L \in Spec_T \mid \forall s \in L \; \exists t \in L: s < t \} .$$

Then P satisfies $(T, \mathcal{L})$ iff P is always able to extend its present communication trace s by some further communication. (Since $T = Comm^*$ there is no restriction in which communications P may participate.) //

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 behaviour as a general liveness property $(T, \mathcal{L})$ with $T = \{a,b\}^*$ and

$$\mathcal{L} = \{ L_n \mid n \geq 0 \} \text{ where } L_n = \{ \varepsilon, a, \ldots, a^n, a^n b \} .$$

Then P satisfies $(T, \mathcal{L})$ if there is some $n \geq 0$ such that P communicates b after n communications a; but it is not known in advance which n applies. //

Example 9.5 Similarly, we can express the concept of a bounded buffer as a general liveness property $(T, \mathcal{L})$ with

$$T = \{ s \mid s \in \{in,out\}^* \wedge out\#s \leq in\#s \} \text{ and}$$
$$\mathcal{L} = \{ L_n \mid n \geq 0 \}$$

where L_n is taken from Example 9.2. Now P satisfies $(T, \mathcal{L})$ 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 $\mathcal{R}$

This section improves the Divergence Model $\mathcal{D}$ into a new model which can deal with simple (and a certain type of general) liveness properties: the Readiness Model $\mathcal{R}$. Moreover, $\mathcal{R}$ allows us to treat now the full language $CRec(\Sigma)$ of Communicating Processes by distinguishing between external nondeterminism $\square$ and internal nondeterminism or. The idea of $\mathcal{R}$ is as follows: we assume that not only the "past" of a process can be observed via traces also a part of the "future" via so-called expectation [12] or ready sets X [20, 16] 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 has ceased (see also Section 13).

Our observations have the form:

- s trace of successful communications,
- sX ready set X presented by the process after s,
- s↑ possibility of divergence.

Thus we get

$$Obs_{\mathcal{R}} = \{ s, sX, s\uparrow \mid s \in Comm^* \wedge X \subseteq Comm \}.$$

Let s,t range over $Comm^*$, X, Y over $\mathcal{P}(Comm)$ and Δ over $\mathcal{P}(Comm) \cup \{\uparrow\}$. The successor relation $\longrightarrow$ is the smallest relation on $Obs_{\mathcal{R}}$ satisfying

$$s \longrightarrow sX,\ s \longrightarrow sa,\ s \longrightarrow s\uparrow$$
$$sX \longrightarrow sb \text{ for all } b \in X$$
$$s\uparrow \longrightarrow sX,\ s\uparrow \longrightarrow sa,\ s\uparrow \longrightarrow sa\uparrow$$

for all $s \in Comm^*$, $X \subseteq Comm$ and $a \in Comm$. Relation $\longrightarrow$ 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. Divergence s↑ is (as in the Divergence Model $\mathcal{D}$) identified with every possible subsequent behaviour.

As extensibility relation $\longrightarrow\!\!\!\!\!\rightarrow$ we take the smallest relation between $Obs_{\mathcal{R}}$ and $\mathcal{P}(Obs_{\mathcal{R}})$ such that

$$s \longrightarrow\!\!\!\!\!\rightarrow sX$$
$$sX \longrightarrow\!\!\!\!\!\rightarrow \{sb \mid b \in X\}$$
$$s\uparrow \longrightarrow\!\!\!\!\!\rightarrow \{sX,\ sa,\ sa\uparrow \mid X \subseteq Comm \wedge a \in Comm\}$$

holds for all $s \in Comm^{*}$ and $X \subseteq Comm$. The process specifications of $\mathcal{R}$ are given by the full specification space $Spec_{\mathcal{R}}$ over $(Obs_{\mathcal{R}}, \longrightarrow, \longrightarrow\!\!\!\!\!\rightarrow)$. Then every $S \in Spec_{\mathcal{R}}$ realises a <u>local liveness principle</u>: every $s \in S$ and $sX \in S$ with $X \neq \emptyset$ requires certain immediate successor observations to be present in S due to $\longrightarrow\!\!\!\!\!\rightarrow$. Only observations

$$s\emptyset$$

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

(1) <u>stop</u> $= \{\varepsilon,\ \varepsilon\emptyset\}$

(2) <u>div</u> $= Obs_{\mathcal{R}}$

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

(4) S_1 <u>or</u> $S_2 = S_1 \cup S_2$

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

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.

$$(a \longrightarrow P) \,\Box\, (b \longrightarrow Q)$$

will have an initial ready set $\{a,b\}$ indicating that the

environment can choose whether the process behaves like $a \longrightarrow P$ or like $b \longrightarrow Q$ by first communicating either a or b. In contrast

$$(a \longrightarrow P) \ \underline{or} \ (b \longrightarrow Q)$$

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

(6) $$S_1 \|_A S_2 = \left\{ s, sX \;\middle|\; \begin{array}{l} \exists t_1 X_1 \in S_1,\ t_2 X_2 \in S_2: \\ \quad s \in t_1 \|_A t_2 \wedge X = X_1[\overline{A}]X_2 \end{array} \right\}$$
$$\cup \left\{ st, st\Delta \;\middle|\; \begin{array}{l} \exists t_1 \in S_1,\ t_2 \in S_2: \\ \quad s \in t_1 \|_A t_2 \wedge (t_1\uparrow \in S_1 \vee t_2\uparrow \in S_2) \end{array} \right\}$$

The first clause of the defintion uses the majority operator of Section 2 for $\overline{A} = Comm \setminus A$:

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

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) $S \setminus b$: we first introduce the hiding relation $g \subseteq Obs_{\mathcal{R}} \times Obs_{\mathcal{R}}$ which describes how observations about S are related to those about $S \setminus b$:

(i)	$s \ g \ s \setminus b$	
(ii)	$sX \ g \ (s \setminus b)X$	provided $b \notin X$
(iii)	$sX \ g \ s \setminus b$	provided $b \in X$
(iv)	$s\uparrow \ g \ s \setminus b$	

for all s,t,X,Δ. Clause (iii) may require a comment: since communication b has become internal in $S \setminus b$, the stable state sX of S with $b \in X$ has become unstable in $S \setminus b$ 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

$sX\ g\ s\setminus b$ provided $b \in X$.

This definition agrees with the decisions taken in [12] and [20].

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

$$S \setminus b = \{\, s\setminus b, (s\setminus b)X \mid sX \in S \wedge b \notin X \,\} \cup \{\, (s\setminus b)t, (s\setminus b)t\Delta \mid \forall n \geq 0 : sb^n \in S \,\}$$

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

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

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

$$S_{\mathcal{R}}(T,L) = T \cup \left\{ sX \;\middle|\; \begin{array}{l} s \in T \wedge s\cdot X \subseteq T \\ \quad \wedge\ (\underline{\text{if}}\ s \in L\ \underline{\text{then}}\ \mathrm{succ}_L(s) \subseteq X) \end{array} \right\}$$

As abbreviations we use here $s\cdot X = \{sa \mid a \in X\}$ and $\mathrm{succ}_L(s) = \{a \mid sa \in L\}$ for $s \in \mathrm{Comm}^*$, $X \subseteq \mathrm{Comm}$ and $L \in \mathrm{Spec}_{\mathcal{T}}$.

We define now

$$P \text{ satisfies } (T,L) \text{ iff } \mathcal{R} \models P\ \underline{\text{sat}}\ S_{\mathcal{R}}(T,L)\ .$$

This definition formalises the intuitive correctness criteria (1) - (3) given in Section 9. This is clear for (1) and (2). Condition (3) is expressed by the clause

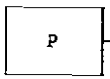
$$\underline{\text{if}}\ s \in L\ \underline{\text{then}}\ \mathrm{succ}_L(s) \subseteq X$$

by which the ready set X of a trace $s \in L$ always includes the required successor communications $a \in \mathrm{succ}_L(s)$. Because of $\varepsilon \in L$ we can in particular start with all initial communications in L. And these communications are independent of internal activities of P since the "readies" sX all refer to "stable states".

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 $\mathcal{R}$. This yields the following specification SEND = $S_{\mathcal{R}}(T,L)$:

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

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

(*) 

ε , c , cc , ccc , ...

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

$$\mathcal{R} \models P \text{ sat SEND}$$

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

$$P = \mu\xi.(c \rightarrow \xi) .$$

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 $\text{BUFF}_n = S_{\mathcal{R}}(T_n, L_n)$:

$$\text{BUFF}_n = \left\{ s, sX \;\middle|\; \begin{array}{l} s \in \{in,out\}^* \wedge out\#s \leq in\#s \leq (out\#s)+n \\ \wedge\ (\underline{\text{if}}\ in\#s < (out\#s)+n\ \underline{\text{then}}\ in \in X) \\ \wedge\ (\underline{\text{if}}\ out\#s < in\#s\ \underline{\text{then}}\ out \in X) \end{array} \right\}$$

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

$$P_1(in,out) = \mu\xi.(in \rightarrow out \rightarrow \xi)$$

and define inductively

$$P_{n+1}(in,out) = (P_1(in,wire) \parallel_{\{wire\}} P_n(wire,out)) \setminus wire$$

Then we can show

$$\mathcal{R} \models P_n(in,out) \text{ sat } BUFF_n \ .$$

Note that differently from Example 6.1

$$BUFF_n \not\subseteq BUFF_{n+1} \ .$$

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

$$R_2 = in \rightarrow \mu\xi . (\quad in \rightarrow out \rightarrow \xi \\ \Box \ out \rightarrow in \rightarrow \xi)$$

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

Next we investigate general liveness properties $(T,\mathcal{L})$. Recall from Section 9 that we define:

P satisfies $(T,\mathcal{L})$ if $\exists L \in \mathcal{L}$: P satisfies (T,L).

We extend this defintion to sets $\mathcal{P}$ of processes by

$\mathcal{P}$ satisfies $(T,\mathcal{L})$ if $\forall P \in \mathcal{P}$: P satisfies $(T,\mathcal{L})$.

Now we introduce the following concept of expressiveness.

Definition 10.3 A liveness property $(T,\mathcal{L})$ is expressible in a specification-oriented model $\mathcal{M}$ if there is a specification $S \in Spec_{\mathcal{M}}$ such that

P satisfies $(T,\mathcal{L})$ iff $\mathcal{M} \models P$ sat S

holds for every process $P \in CRec(\Sigma)$. We also say that S expresses $(T,\mathcal{L})$ in $\mathcal{M}$.

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

Example 10.4 The introduction of observations sØ 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

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

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 livness property $(T,\mathcal{L})$ of Example 9.3. //

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

Proposition 10.5 $(T,\mathcal{L})$ is expressible in $\mathcal{R}$ iff the following holds for all $\mathcal{P} \subseteq \text{CRec}(\Sigma)$ and $Q \in \text{CRec}(\Sigma)$: whenever $\mathcal{P}$ satisfies $(T,\mathcal{L})$ and

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

then also Q satisfies $(T,\mathcal{L})$.

Proof. "only if": by the definition of sat.
"if": let $\mathcal{P} = \{\, P \mid P \text{ satisfies } (T,\mathcal{L}) \,\}$.
Case 1: $\mathcal{P} = \emptyset$. Then there is no $L \in \mathcal{L}$ and no process P such that P satisfies (T,L). Thus for arbitrary $L \in \mathcal{L}$ the specification $S = S_{\mathcal{R}}(T,L)$ expresses $(T,\mathcal{L})$.
Case 2: $\mathcal{P} \neq \emptyset$. Then $S = \bigcup_{P \in \mathcal{P}} \mathcal{R}[\![P]\!]$ expresses $(T,\mathcal{L})$. Note that $S \in \text{Spec}_{\mathcal{R}}$.
//

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

$$Q = \mu\xi.(a \rightarrow \xi) \text{ and } P_n = \underbrace{a \rightarrow \ldots \rightarrow a}_{n} \rightarrow b$$

for $n \geqslant 0$. Then P_n satisfies $(T,\mathcal{L})$ and $\mathcal{R}[\![Q]\!] \subseteq \bigcup_n \mathcal{R}[\![P_n]\!]$, but Q does not satisfy $(T,\mathcal{L})$. Thus $(T,\mathcal{L})$ is not expressible in $\mathcal{R}$ by Proposition 10.5. This limitation in expressiveness is typical for any kind of finitary observation (see Section 13), not only for readies sX. 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 $(T,\mathcal{L})$ of Example 9.5. Clearly

$$P_n(\text{in},\text{out}) \text{ satisfies } (T,\mathcal{L})$$

by the previous Example 10.2. Now consider

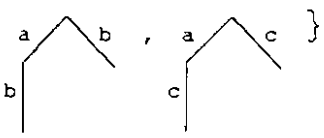
$$P_\infty = \mu\, \xi\, .(\text{in} \rightarrow (\xi \parallel_\emptyset \text{out} \rightarrow \underline{\text{stop}}))\ .$$

P_∞ expresses an infinite buffer:

$$\mathcal{R}[\![P_\infty]\!] = \left\{ s,\ sX \;\middle|\; \begin{array}{l} s \in \{\text{in},\text{out}\}^* \wedge\ \text{out}\#s \leq \text{in}\#s \\ \qquad \wedge\ \text{in} \in X \\ \qquad \wedge\ (\underline{\text{if}}\ \text{out}\#s < \text{in}\#s\ \underline{\text{then}}\ \text{out} \in X) \end{array} \right\}$$

Thus $\mathcal{R}[\![P_\infty]\!] \subseteq \bigcup_n \mathcal{R}[\![P_n(\text{in},\text{out})]\!]$ but P_∞ does not satisfy $(T,\mathcal{L})$. Hence $(T,\mathcal{L})$ is not expressible in $\mathcal{R}$. 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 $\mathcal{R}$. Take e.g. $(T,\mathcal{L})$ with $T = \text{Comm}^*$ and

$\mathcal{L} = \{$  $\}$

(Here we use an equivalent tree notation for prefix-closed sets of trees.) The idea of $\mathcal{L}$ is that b (or c) is possible after a only if it was also possible earlier as an alternative to a.

Consider now

$$P_1 = (a \longrightarrow b \longrightarrow \underline{stop}) \,\square\, b \longrightarrow \underline{stop}$$

$$P_2 = (a \longrightarrow c \longrightarrow \underline{stop}) \,\square\, c \longrightarrow \underline{stop}$$

$$Q = (a \longrightarrow c \longrightarrow \underline{stop}) \,\square\, b \longrightarrow \underline{stop}$$

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

This limitation in expressiveness is typical for trace-like observations like readies sX. 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. //

Next we relate the models $\mathcal{R}$ and $\mathcal{D}$. Let $g \subseteq Obs_{\mathcal{R}} \times Obs_{\mathcal{D}}$ be the projection

$$sX \; g \; t \text{ iff } s = t$$

for observations sX and the identity otherwise. Then the pointwise extension Θ_g satisfies $\Theta_g(S) \in Spec_{\mathcal{D}}$ for every $S \in Spec_{\mathcal{R}}$.

Proposition 10.7 For every process $P \in CRec(\Sigma 2)$ the equation $\Theta_g(\mathcal{R}[\![P]\!]) = \mathcal{D}[\![P]\!]$ holds.

Proof. By Propositions 2.1 and 5.2. //

We conclude with some comments on related work. As already indicated, the idea of ready sets is taken from [12,16,20], but the details of the Readiness Model $\mathcal{R}$ are new. In particular [12] does not abstract from internal activities: little δ's denoting internal progress remain in their traces. It is interesting to note that the model $\mathcal{R}$ is well suited as a basis for implementing processes in a functional style [38].

A restricted version of a Readiness Model forms also a basis of [31]. Essentially [31] use only a liveness principle of the form

$$\forall x \in Obs - Max \; \exists y \in S: \; x \longrightarrow\!\!\!\!\triangleright \; y$$

whereas in $\mathcal{R}$ ready sets can require more than one successor of an observation (trace) to be present (cf. the introduction of $\longrightarrow\!\!\!\!\!\triangleright$ in Section 8). Consequently [31] cannot deal with external nondeterminism.

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

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

$$\mathcal{M}[\![P]\!] = \bigcap \{ \mathcal{M}[\![Q_{div}]\!] \mid P \stackrel{*}{\longmapsto} Q \}$$

(cf. Section 2). Therefore finite processes can be considered as an important tool for reasoning about general processes (cf. e.g. the proof of Theorem 8.4.).

This reasoning is simplified very much if further on every finite process can be reduced to a so-called primitive finite process $P \in FRec(\Sigma_p)$ where $\Sigma_p \subseteq \Sigma$ with

$$Op(\Sigma_p) = \{ \underline{stop}, \underline{div} \} \cup \{ a \rightarrow \mid a \in Comm \} \cup \{ \underline{or}, \Box \}$$

does not involve parallelism $\|_A$ or hiding $\backslash b$. Therefore we would like to have models $\mathcal{M}$ which admit reduction in the following sense:

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

$$F = P$$

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

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

is true in $\mathcal{M}$.

Clearly $\mathcal{M}$ admits reduction from $FRec(\Sigma)$ to $FRec(\Sigma_p)$ iff every operator $f \in Op(\Sigma)$ is reducible to $FRec(\Sigma_p)$ in $\mathcal{M}$. Unfortunately, our previous Readiness Model $\mathcal{R}$ does not admit reduction to $FRec(\Sigma_p)$. The troublesome operator in $\mathcal{R}$ is hiding $\backslash b$. To see this let us study an example where

$$(P \,\square\, Q) \setminus b$$

cannot be expressed without $\setminus b$.

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

(1) $\quad sa \in \mathcal{R}[\![P]\!]$ implies $\exists X \subseteq Comm: a \in X \wedge sX \in \mathcal{R}[\![P]\!]$

i.e. every communication a that can occur while the process is running can also occur after the process has reached stability. Consider now

$$P = a \rightarrow \underline{stop} \quad \text{and} \quad Q = b \rightarrow c \rightarrow \underline{stop} \ .$$

Then we get

$$\mathcal{R}[\![(P \,\square\, Q) \setminus b]\!] = \{ \varepsilon, a, a\emptyset, \varepsilon\{c\}, c, c\emptyset \}$$

which does not satisfy (1). Thus $(P \,\square\, Q) \setminus b$ is not representable in $FRec(\Sigma p)$.

What would be a good candidate in $FRec(\Sigma p)$ to represent this process? We suggest

$$P1 = (a \rightarrow \underline{stop} \,\square\, c \rightarrow \underline{stop}) \ \underline{or} \ (c \rightarrow \underline{stop})$$

with $\mathcal{R}[\![P1]\!] = \mathcal{R}[\![(P \,\square\, Q) \setminus b]\!] \cup \{\varepsilon\{a,c\}\}$ because P1 and $(P \,\square\, Q) \setminus b$ satisfy in $\mathcal{R}$ exactly the same simple liveness specifications $S_{\mathcal{R}}(T,L)$. Indeed for T,L as in Section 9

$$\mathcal{R} \models (P \,\square\, Q) \setminus b \ \underline{sat} \ S_{\mathcal{R}}(T,L)$$

iff $T \supseteq \{ \varepsilon, a, c \}$ and $L \subseteq \{ \varepsilon, c \}$. But these are exactly the sets T,L with

$$\mathcal{R} \models P1 \ \underline{sat} \ S_{\mathcal{R}}(T,L) \ .$$

Thus identifying

(2) $\quad (P \,\square\, Q) \setminus b = P1$

does not affect the expressive power of our model in terms of simple liveness properties (see also Theorem 11.8). //

We now explain a model which admits reduction to FRec(Σp) essentially since the suggested law (2) is true: the Refusal or Failure Model $\mathcal{F}$ based on [22,39,7]. The initial idea of $\mathcal{F}$ looks quite different from $\mathcal{R}$. 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

$$Obs_{\mathcal{F}} = \{ sX, s\uparrow \mid s \in Comm^* \wedge X \subseteq Comm \}.$$

Observations sX are called failures and the sets X refusal sets [22]. As in Section 10 we let s,t range over Comm, X,Y over $\mathcal{P}$(Comm) and Δ over $\mathcal{P}(Comm) \cup \{\uparrow\}$. As successor relation $\longrightarrow$ we take the smallest relation on $Obs_{\mathcal{F}}$ satisfying

$$s\emptyset \longrightarrow sa\emptyset \;,\; s\emptyset \longrightarrow sX \;,\; s\emptyset \longrightarrow s\uparrow$$

$$s\uparrow \longrightarrow sX \;,\; s\uparrow \longrightarrow sa\emptyset \;,\; s\uparrow \longrightarrow sa$$

for all $s \in Comm^*$, $a \in Comm$ and $X \subseteq Comm$ with $X \neq \emptyset$. Here failures $s\emptyset$ represent interactions where no communication has been refused so far. As in the previous models $\mathcal{D}$ and $\mathcal{R}$ divergence $s\uparrow$ is identified with s followed by every possible subsequent behaviour.

The dynamic aspect of (*) is captured by the following extensibility relation $\longrightarrow\!\!\!\!\!\rightarrow$ between $Obs_{\mathcal{F}}$ and $\mathcal{P}(Obs_{\mathcal{F}})$:

$s\emptyset \longrightarrow\!\!\!\!\rightarrow$ SUCC iff $\forall X \subseteq Comm$: ($\exists a \in X$: $sa\emptyset \in SUCC$

$\vee\ \forall Y \subseteq X$ with $Y \neq \emptyset$: $sY \in SUCC$)

$s\uparrow \longrightarrow\!\!\!\!\rightarrow \{sX,\ sa\emptyset,\ sa\uparrow \mid X \subseteq Comm \text{ with } X \neq \emptyset \wedge a \in Comm\}$

Let us explain the more complex clause $s\emptyset \longrightarrow\!\!\!\!\rightarrow$ 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 $X \subseteq Comm$ of communications 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 $\emptyset \neq Y \subseteq X$ ($Y \neq \emptyset$ guarantess $s\emptyset \longrightarrow sY$). Note that this definition reflects the informal description (*) above.

As process specifications of $\mathcal{F}$ we take the full specification space $Spec_{\mathcal{F}}$ over $(Obs_{\mathcal{F}}, \longrightarrow, \longrightarrow\!\!\!\!\rightarrow)$.

Remark 11.3 A subset $S \subseteq Obs_{\mathcal{F}}$ is a process specification in $Spec_{\mathcal{F}}$ iff the following holds:

(i) $\varepsilon\emptyset \in S$
(ii) $st\emptyset \in S$ implies $s\emptyset \in S$
(iii) $sX \in S \wedge Y \subseteq X$ implies $sY \in S$
(iv) $sX \in S \wedge sa\emptyset \notin S$ implies $s(X \cup \{a\}) \in S$
(v) $s\uparrow \in S$ implies $st\Delta \in S$ for all t, Δ.

The Failure Model $\mathcal{F}$ consists of $Spec_{\mathcal{F}}$ and the following set $\{f_{\mathcal{F}} \mid f \in Op(\Sigma)\}$ of operators (again we drop the index $\mathcal{F}$):

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

The deadlocking process can refuse any set X.

(2) div $= Obs_{\mathcal{F}}$

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

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

(4) S_1 or $S_2 = S_1 \cup S_2$

$$(5)\quad S_1 \,\square\, S_2 = \{\varepsilon X \mid \varepsilon X \in S_1 \cap S_2\} \cup \{\varepsilon\Delta \mid \varepsilon\uparrow \in S_1 \cup S_2\} \cup \{s\Delta \mid s \neq \varepsilon \wedge s\Delta \in S_1 \cup S_2\}$$

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

$$(6)\quad S_1 \,\|_A\, S_2 = \left\{ sX \;\middle|\; \begin{array}{l} \exists t_1 X_1 \in S_1,\ t_2 X_2 \in S_2: \\ \quad s \in t_1 \,\|_A\, t_2 \wedge X = X_1[A]X_2 \end{array} \right\} \cup \left\{ st\Delta \;\middle|\; \begin{array}{l} \exists t_1\emptyset \in S_1,\ t_2\emptyset \in S_2: \\ \quad s \in t_1 \,\|_A\, t_2 \wedge (t_1\uparrow \in S_1 \vee t_2\uparrow \in S_2) \end{array} \right\}$$

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.

$$(7)\quad S \setminus b = \{ (s \setminus b)X \mid s(X \cup \{b\}) \in S \} \cup \{ (s \setminus b)t\Delta \mid \forall n \geq 0:\ sb^n\emptyset \in S \}$$

Note that $S \setminus b$ can refuse a set X only if $S \setminus b$ 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 Section 5.

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

$$sX\ g\ sZ \text{ iff } Z \subseteq \bar{X}$$

$$s\uparrow\ g\ s\uparrow$$

which interprets sets Z as the downward closures of the complements $\overline{X} = Comm \setminus X$ of ready sets X. By Remark 11.3 the pointwise extension $\mathcal{O}_g$ maps every $S \in Spec_{\mathcal{R}}$ into a process specification $\mathcal{O}_g(S) \in Spec_{\mathcal{F}}$.

Proposition 11.4 For every process $P \in CRec(\Sigma)$ the equation $\mathcal{O}_g(\mathcal{R}[\![P]\!]) = \mathcal{F}[\![P]\!]$ holds.

Proof. To apply Proposition 2.1 we have to show that $\mathcal{O}_g$ is a (strict and) $\supseteq$-continuous homomorphism from $\mathcal{R}$ to $\mathcal{F}$. Domain finiteness of g implies the $\supseteq$-continuity of $\mathcal{O}_g$ by Proposition 5.2. Checking the homomorphism property of $\mathcal{O}_g$ boils down to a simple calculation with downward closures of complements. For example the crucial argument to show

$$\mathcal{O}_g(S_1 \|_{A\,\mathcal{R}} S_2) = \mathcal{O}_g(S_1) \|_{A\,\mathcal{F}} \mathcal{O}_g(S_2)$$

for all $S_1, S_2 \in Spec_{\mathcal{R}}$ is as follows:

$$Z \subseteq \overline{X_1[\bar{A}]X_2} \quad \text{iff} \quad \exists\, Z_1 \subseteq \overline{X}_1 \;\; \exists\, Z_2 \subseteq \overline{X}_2 : Z = Z_1[A]Z_2$$

//

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

Definition 11.5 A process specification $S \in Spec_{\mathcal{R}}$ of the Readiness Model $\mathcal{R}$ is called convex closed [8,33] if the following holds:

(i) $sX, sZ \in S$ implies $s(X \cup Z) \in S$
(ii) $sX, sZ \in S$ and $X \subseteq Y \subseteq Z$ imply $sY \in S$.

For $S \in Spec_{\mathcal{R}}$ let con(S) denote w.r.t. $\supseteq$ strongest convex closed specification with $S \subseteq con(S)$. Clearly $con(S) \in Spec_{\mathcal{R}}$ holds.

Lemma 11.6 For $S_1, S_2 \in Spec_{\mathcal{R}}$ the following holds:
$\mathcal{O}_g(S_1) \subseteq \mathcal{O}_g(S_2)$ iff $con(S_1) \subseteq con(S_2)$.

Proof. By the properties of downward closures of ready sets. //

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

Proposition 11.7 In general $\mathcal{R}$ is more expressive than $\mathcal{F}$ in the following sense:

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

(*) $\quad \mathcal{R} \models P \underline{\text{sat}} S_{\mathcal{R}}$ iff $\mathcal{F} \models P \underline{\text{sat}} S_{\mathcal{F}}$

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

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

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

Proof. (i) Take $S_{\mathcal{R}} = \{s, sX \mid s\emptyset \in S_{\mathcal{F}} \wedge \forall a \in X: s\langle a \rangle \notin S_{\mathcal{F}}\}$. Then $S_{\mathcal{R}} \in \mathrm{Spec}_{\mathcal{R}}$, $S_{\mathcal{R}}$ is convex closed and $\mathcal{O}_g(S_{\mathcal{R}}) = S_{\mathcal{F}}$. Thus $\mathcal{F}[\![P]\!] \subseteq S_{\mathcal{F}}$ iff $\mathcal{O}_g(\mathcal{R}[\![P]\!]) \subseteq \mathcal{O}_g(S_{\mathcal{R}})$ iff $\mathcal{R}[\![P]\!] \subseteq S_{\mathcal{R}}$ (by Proposition 11.4 and Lemma 11.6). Hence (*) holds.
(ii) Consider $P = a \rightarrow \underline{\text{stop}}$, $Q = b \rightarrow \underline{\text{stop}}$, and $S_{\mathcal{R}} = \mathcal{R}[\![P \underline{\text{ or }} Q]\!]$. Suppose (*) holds for some $S_{\mathcal{F}} \in \mathrm{Spec}_{\mathcal{F}}$. Because $\mathcal{F}[\![P \,\Box\, Q]\!] \subseteq \mathcal{F}[\![P \underline{\text{ or }} Q]\!]$ we get $\mathcal{F} \models P \,\Box\, Q \underline{\text{ sat }} S_{\mathcal{F}}$. But $\mathcal{R} \models P \,\Box\, Q \underline{\text{ sat }} S_{\mathcal{R}}$ is false. Contradiction.
(iii) Take $S_{\mathcal{F}} = \mathcal{O}_g(S_{\mathcal{R}})$. Then (*) follows as in (i) from Proposition 11.4 and Lemma 11.6. //

Theorem 11.8 A liveness property $(T, \mathcal{L})$ is expressible in $\mathcal{R}$ iff $(T, \mathcal{L})$ is expressible in $\mathcal{F}$.

Proof. "if": by Proposition 11.7, (i).
"only if": Let $(T, \mathcal{L})$ be expressible in $\mathcal{R}$. Due to Proposition 11.7, (iii) it suffices to show that $(T, \mathcal{L})$ is also expressible by a convex closed specification $S \in \mathrm{Spec}_{\mathcal{R}}$. To see this we re-examine the proof of Proposition 10.5. Consider

$$\mathcal{P} = \{P \mid P \text{ satisfies } (T, \mathcal{L})\}.$$

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

Case 2: $\mathcal{P} \neq \emptyset$. Then $S = \bigcup_{P \in \mathcal{P}} \mathcal{R}[\![P]\!]$ expresses $(T,\mathcal{L})$ due to Proposition 10.5. We show that S is convex closed.

Let $sX, sZ \in S$ and $X \subseteq Y \subseteq Z$. By the defintion of S, there exist $P,Q \in \mathcal{P}$ with $sX \in \mathcal{R}[\![P]\!]$ and $sZ \in \mathcal{R}[\![Q]\!]$. Proposition 10.5 implies $P \underline{\text{ or }} Q \in \mathcal{P}$. Thus there is some $L \in \mathcal{L}$ with $\mathcal{R} \models P \underline{\text{ or }} Q \underline{\text{ sat }} S_{\mathcal{R}}(T,L)$, i.e. with $\mathcal{R}[\![P \underline{\text{ or }} Q]\!] \subseteq S_{\mathcal{R}}(T,L)$. Since $S_{\mathcal{R}}(T,L)$ is convex closed, also

$$con(\mathcal{R}[\![P \underline{\text{ or }} Q]\!]) \subseteq S_{\mathcal{R}}(T,L).$$

Clearly $s(X \cup Z)$, $sY \in con(\mathcal{R}[\![P \underline{\text{ or }} Q]\!])$. Note that there is some process R with

$$s(X \cup Z),\ sY \in \mathcal{R}[\![R]\!] \subseteq con(\mathcal{R}[\![P \underline{\text{ or }} Q]\!]).$$

Because of $\mathcal{R}[\![R]\!] \subseteq S_{\mathcal{R}}(T,L)$ we get $R \in \mathcal{P}$. Thus $s(X \cup Z)$, $sY \in S$. This proves the convex closure of S.

//

Next we turn to the original question of reducing finite processes to primitive ones. The crucial advantage of $\mathcal{F}$ over $\mathcal{R}$ is the following algebraic law of $\mathcal{F}$:

(+) $\quad (P \,\square\, b \longrightarrow Q) \setminus b = (P \,\square\, Q) \setminus b \underline{\text{ or }} Q \setminus b$.

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

(i) $\square$ is commutative and associative; it has a unit $\underline{\text{stop}}$ and a zero $\underline{\text{div}}$, i.e.

$$P \,\square\, \underline{\text{stop}} = P \quad \text{and} \quad P \,\square\, \underline{\text{div}} = \underline{\text{div}}$$

(ii) $\|_A$ is commutative and has a zero $\underline{\text{div}}$:

$$P \,\|_A\, \underline{\text{div}} = \underline{\text{div}}$$

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

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

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

$$a \rightarrow (P \ \underline{or} \ Q) = (a \rightarrow P) \ \underline{or} \ (a \rightarrow Q)$$

$$(P \ \underline{or} \ Q) \Box R = (P \Box R) \ \underline{or} \ (Q \Box R)$$

$$(P \ \underline{or} \ Q) \backslash b = P \backslash b \ \underline{or} \ Q \backslash b$$

$$(P \ \underline{or} \ Q) \|_A R = (P \|_A R) \ \underline{or} \ (Q \|_A R)$$

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

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

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

Proposition 11.9 Parallel composition $\|_A$ is reducible to FRec(Σ_P) in $\mathcal{R}$ and thus in $\mathcal{F}$.

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

$$\underline{stop} \|_A \underline{stop} = \underline{stop}$$

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

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

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

in $\mathcal{R}$. //

Proposition 11.10 Hiding $\setminus b$ is reducible to FRec(Σp) in $\mathcal{F}$.

Proof. By structural induction. For P = stop reducibility of $\setminus b$ follows from the law

$$\text{stop} \setminus b = \text{stop}$$

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

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

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

$$P \setminus b = \Box_{a \in A}\ a \longrightarrow (P_a \setminus b)$$

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

The previous propositions are summarised in:

Corollary 11.11 The Failure Model $\mathcal{F}$ admits reduction from FRec(Σ) to FRec(Σp).

The Failure Model originally proposed in [22] has recently attracted much attention in the literature. Whereas our Failure Model $\mathcal{F}$ can be considered as a refinement of the Divergence Model $\mathcal{D}$, the original model in [22] is a refinement of the Trace Model $\mathcal{T}$ discussed in Section 7. Consequently the problems concerning divergence signalled in Section 8 are also present in the original model [22]. This was first realised independently in work of [39,7,32].

Our present model $\mathcal{F}$ is closest to the one proposed in [39] and isomorphic to the one developed in [7] where also a complete proof system for semantic equality of finite processes is given. This proof system uses some additional algebraic laws to the ones needed here to prove reducibility of $\mathcal{F}$. Closely related to the Failure Model $\mathcal{F}$ are also the models produced in [33] by starting from a general notion of testing related to ideas of [25]. Models

combining aspects of $\mathcal{R}$ and $\mathcal{F}$ have been investigated in [40].

The main difference between our approach to $\mathcal{F}$ and the previous research just cited is that we have presented $\mathcal{F}$ here as a special example in the general setting of specification-oriented semantics. Together with the series of models $\mathcal{C}$, $\mathcal{T}$, $\mathcal{D}$ and $\mathcal{R}$ we hope this gives a better insight into the structure of $\mathcal{F}$ 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 design decisions (cf. e.g. the idea of a "stable state" in Section 9). 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 [24,27,37]. 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

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

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

(1) stop has no transition.

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

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

(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 $\lambda \in Act$:

If $P \xrightarrow{\lambda} P1$ then $(P + Q) \xrightarrow{\lambda} P1$ and $(Q + P) \xrightarrow{\lambda} P1$ [27].

The reason for choosing $\square$ rather than + is that $\square$ avoids a number of complications encountered with + (see e.g. [27, Chap. 7]).

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

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

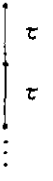
(8) $\mu \xi . P \xrightarrow{\tau} P[\mu \xi . P / \xi]$.

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

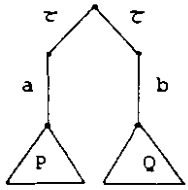
It is sometimes helpful to visualise the possible transitions of a process by so-called synchronisation trees [27]. These are rooted, unordered trees whose arcs are labelled with actions $\lambda \in Act$. We show some typical cases.

Example 12.1

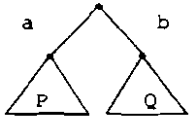
(i) $\underline{div} =$



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

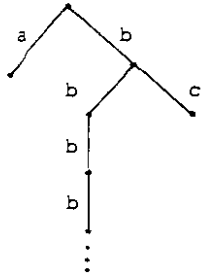


(iii) $(a \rightarrow P) \,\square\, (b \rightarrow Q)$ =

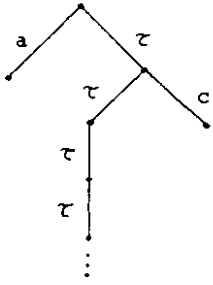


(iv) Hiding b in a synchronisation tree P simply means relabelling all arcs b into τ:

P =



P \ b =



Transitions $\xrightarrow{\lambda}$ are extended in two ways. For words $w = \lambda_1 \ldots \lambda_n \in Act^*$ 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 $s \in Comm^*$ we write

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

if there exists some $w \in Act^*$ with $P \xrightarrow{w} Q$ and $s = w \setminus \tau$ where $w \setminus \tau$ denctes 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 <u>diverges at s</u> if

$$\exists Q \ \forall n \geq 0 \ \exists R: \ P \overset{s}{\Longrightarrow} Q \wedge Q \xrightarrow{\tau^n} R \ .$$

P is <u>divergence free</u> 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.)

<u>Example 12.2</u> (i) $a \rightarrow \underline{div}$ diverges at a.
(ii) <u>div</u>, $\mu \xi \ . \ \xi$ and $(\mu \xi \ . b \rightarrow \xi) \setminus b$ all diverge at ε. //

Finally, we introduce a modification of Milner's <u>observational equivalence</u> $\approx$ [17,27,28] which takes the notion of divergence into account. $\approx$ is defined by the following series of equivalence relations $\approx_l$, $l \geq 0$, over $CRec(\Sigma)$:

$P \approx_0 Q$ 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 $\forall s \in Comm^*$ with $|s| \leq l$:
(i) $P \overset{s}{\Longrightarrow} P1$ implies $\exists Q1: \ Q \overset{s}{\Longrightarrow} Q1 \wedge P1 \approx_l Q1$
(ii) $Q \overset{s}{\Longrightarrow} Q1$ implies $\exists P1: \ P \overset{s}{\Longrightarrow} P1 \wedge Q1 \approx_l Q1$

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

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 $\lambda_i \in Comm$. Since this does not exclude branches with arbitrarily many internal actions τ, it is in general impossible to establish $P \approx_l Q$ effectively.

<u>Example 12.3</u> (i) $(b \rightarrow P) \setminus b \approx P \setminus b$
(ii) $((b \rightarrow P) \setminus b) \ \square \ Q \approx (P \setminus b) \ \square \ Q$
(iii) <u>div</u> $\approx \mu \xi \ . \ \xi$, but <u>div</u> $\not\approx_1$ <u>stop</u> <u>or</u> <u>div</u>.
(iv) $a \rightarrow (P \ \underline{or} \ Q) \not\approx_2 (a \rightarrow P) \ \underline{or} \ (a \rightarrow Q)$

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

$$\underline{\text{div}} = \underline{\text{stop}}\ \underline{\text{or}}\ \underline{\text{div}} \quad \text{and}$$
$$a \longrightarrow (P\ \underline{\text{or}}\ Q) = (a \longrightarrow P)\ \underline{\text{or}}\ (a \longrightarrow Q)$$

hold in all models $\mathcal{C}$, $\mathcal{T}$, $\mathcal{D}$, $\mathcal{R}$ and $\mathcal{F}$ (cf. Sections 8 and 11). The precise relationship between our denotational models and the operational semantics will be discussed in 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 require that $\Vdash$ agrees with the observational equivalence $\approx$ introduced in the previous section:

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

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

holds.

Informally, this defintion 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 defintion 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 actions τ in synchronisation trees of processes P are not detectable by observations: cf. Example 12.3, (i) and (ii).

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

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

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

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

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

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

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

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

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

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

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

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

A_m: $\mathcal{M}[\![P]\!] = Obs[\![P]\!]$
holds for every $P \in CRec(\Sigma O)$ with at most m occurrences of μ.

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

C_m: $Obs[\![\mu\xi.P(\overline{Q})]\!] = \bigcap_{n \to 0} Obs[\![(P(\overline{Q}))^n(\underline{div})]\!]$
holds for every $P \in Rec(\Sigma)$ 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 $\overline{Q} = Q_1,\ldots,Q_r$ of processes in $CRec(\Sigma O)$ is substituted to yield $P(\overline{Q})$ with at most ξ as free identifier. To ξ an n-fold substitution starting with $\underline{div}$ is applied to yield $(P(\overline{Q}))^n(\underline{div})$.

We wish to show that A_m holds for every $m \geq 0$. 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 (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 implies B_{m+1} ,

C_m and A_m together imply A_{m+1} .

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

B_m implies C_m .

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

Case 1: $\mu\xi.P$ diverges at ε .

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

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

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

(*) $\quad Obs[\![\mu\xi.P]\!] = Obs[\![P^n(\mu\xi.P)]\!]$

holds for every $n \geq 0$. For $S \subseteq Obs$ and $l \geq 0$ we define

$$l{:}S = \{x \in S \mid \text{where } \|x\| = l\} .$$

We show that

(**) $\quad l{:}Obs[\![P^n(R_1)]\!] = l{:}Obs[\![P^n(R_2)]\!]$

holds for every $n \geq l+1$ and all $R_1, R_2 \in CRec(\Sigma)$:

Take some $R \in CRec(\Sigma)$ and consider an observation $x \in l{:}Obs[\![P^n(R)]\!]$. Since $\mu\xi.P$ does not diverge at ε , there is no transition chain

$$P(R) \xrightarrow{\lambda_1 \cdots \lambda_k} R$$

with all $\lambda_i = \tau$ (and the R on the RHS denotes that occurrence of R which was substituted for ξ in P on the LHS). Thus at least one λ_i is an observable communication, say $\lambda_i = a \in Comm$. In other words: $\mu\xi.P$ is an instance of a guarded recursion. Consequently, every transition chain

$$P^n(R) \xrightarrow{\lambda_1 \cdots \lambda_k} R$$

needs at least n observable $\lambda_i \in Comm$ to reach R. Hence $P^n(R_1) \approx_1 P^n(R_2)$ holds. This implies (**).

We can now verify C_m:

$$Obs[\![\mu\xi.P]\!] = \bigcup_{l \geq 0} l{:}Obs[\![\mu\xi.P]\!] \quad \text{(definition 1:S)}$$

$$= \bigcup_{l \geq 0} l{:}\Big(\bigcap_{n \geq l+1} Obs[\![P^n(\mu\xi.P)]\!] \Big) \quad \text{(by (*))}$$

$$= \bigcup_{l \geq 0} \Big(\bigcap_{n \geq l+1} l{:}\, Obs[\![P^n(\mu\xi.P)]\!] \Big) \quad \text{(definition 1:S)}$$

$$= \bigcup_{l \geq 0} \Big(\bigcap_{n \geq l+1} l{:}\, Obs[\![P^n(\underline{div})]\!] \Big) \quad \text{(by (**))}$$

$$= \bigcup_{l \geq 0} l{:}\Big(\bigcap_{n \geq l+1} Obs[\![P^n(\underline{div})]\!] \Big) \quad \text{(definition 1:S)}$$

$$= \bigcup_{l \geq 0} l{:}\Big(\bigcap_{n \geq 0} Obs[\![P^n(\underline{div})]\!] \Big) \quad \text{(by } B_m \text{ and (2))}$$

$$= \bigcap_{n \geq 0} Obs[\![P^n(\underline{div})]\!] \quad \text{(definition 1:S)}$$

This finishes our proof. //

We apply now Theorem 13.3 to relate our most detailed model, the Readiness Model $\mathcal{R}$, with the transition semantics. First we need two auxiliary notions for processes P based on the transition structure:

$$next(P) = \{ \lambda \mid \exists Q\colon P \xrightarrow{\lambda} Q \}$$

P is stable iff $\tau \notin next(P)$

This explains the notion of stability used earlier in Section 10.

Now we can define a logical structure $\Vdash_{\mathcal{R}}$ for $Obs_{\mathcal{R}}$:

$$P \Vdash_{\mathcal{R}} t\uparrow \text{ iff } \exists\, s \leq t\colon P \text{ diverges at } s.$$

$$P \Vdash_{\mathcal{R}} t \text{ iff } \exists\, Q\colon P \xRightarrow{t} Q \vee P \Vdash_{\mathcal{R}} t\uparrow$$

$$P \Vdash_{\mathcal{R}} tX \text{ iff } (\exists\, Q\colon P \xRightarrow{t} Q \wedge Q \text{ stable} \wedge \text{next}(Q) = X) \vee (P \Vdash_{\mathcal{R}} t\uparrow)$$

Proposition 13.4 The Readiness Model $\mathcal{R}$ is consistent w.r.t. $\Vdash_{\mathcal{R}}$.

Proof. Condition (2) of Theorem 13.3 holds by the definition of $\Vdash_{\mathcal{R}}$. For condition (1) let us present the crucial facts for two of the more interesting cases: $\Box$ and $\setminus b$.

(a) Crucial for proving

$$Obs_{\mathcal{R}}[\![P \Box Q]\!] = Obs_{\mathcal{R}}[\![P]\!] \Box_{\mathcal{R}} Obs_{\mathcal{R}}[\![Q]\!]$$

are the following two assertions:

(i) $P \Box Q$ diverges at s iff P diverges at s or Q diverges at s.

(ii) $P \Box Q \xRightarrow{\varepsilon} R \wedge R$ stable $\wedge$ next$(R) = X$

iff $\exists\, R_1, R_2, X_1, X_2\colon P \xRightarrow{\varepsilon} R_1 \wedge Q \xRightarrow{\varepsilon} R_2 \wedge R_1, R_2$ 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

$$Obs_{\mathcal{R}}[\![P \setminus b]\!] = Obs_{\mathcal{R}}[\![P]\!] \setminus_{\mathcal{R}} b$$

we need the following:

(i) $P \setminus b$ diverges at t

iff $\exists\, s\ \exists\, Q\colon s \setminus b = t \wedge P \xRightarrow{s} Q$

$\wedge$ (Q diverges at $\varepsilon \vee \forall\, n > 0\ \exists\, R\colon Q \xRightarrow{b^n} R$)

(ii) $P \setminus b \xRightarrow{t} R \wedge R$ stable $\wedge$ next$(R) = X$

iff $\exists\, s\ \exists\, Q\colon s \setminus b = t \wedge Q \setminus b = R \wedge P \xRightarrow{s} Q$

$\wedge\, Q$ stable $\wedge\, b \notin$ next$(Q) \wedge$ next$(Q) = X$.

Then we get

$$Obs_{\mathcal{R}}[\![P \setminus b]\!] = \{ s\setminus b, (s\setminus b)X \mid sX \in Obs_{\mathcal{R}}[\![P]\!] \wedge b \notin X \} \cup \{ (s\setminus b)t\,\Delta \mid s\uparrow \in Obs_{\mathcal{R}}[\![P]\!] \vee \forall n \geq 0:\ sb^n \in Obs_{\mathcal{R}}[\![P]\!] \}$$

The disjunct "$s\uparrow \in Obs_{\mathcal{R}}[\![P]\!]$" can be removed since it implies $Obs_{\mathcal{R}}[\![P]\!] = Obs_{\mathcal{R}}$ (by condition (2)) and thus also $\forall n \geq 0:\ sb^n \in Obs_{\mathcal{R}}[\![P]\!]$. Hence (b) follows.

The remaining operators require similar arguments. //

(Weak) consistency of the more abstract models can be stated as corollaries of Proposition 13.4.

Corollary 13.5 The Failure Model $\mathcal{F}$ and the Divergence Model $\mathcal{D}$ are both consistent.

Proof. Propositions 11.4 and 10.5 explain how to modify $\vdash_{\mathcal{R}}$ to obtain consistent logical structures $\vdash_{\mathcal{F}}$ and $\vdash_{\mathcal{D}}$ for $\mathcal{F}$ and $\mathcal{D}$. //

The Trace Model $\mathcal{T}$, however, is not fully consistent with the transition semantics. Already when introducing the Divergence Model $\mathcal{D}$ in Section 8 we argued that the law

$$(*) \qquad \underline{div} \parallel_{Comm} P = P$$

of $\mathcal{T}$ looks unrealistic. Indeed (*) is the reason for $\mathcal{T}$'s inconsistency. To see this look at the example $P = \underline{stop}$. Since

$$\underline{div} \parallel_{Comm} \underline{stop} \xrightarrow{\tau} \underline{div} \parallel_{Comm} \underline{stop}$$

is the only transition of $\underline{div} \parallel_{Comm} \underline{stop}$, no logical structure $\vdash$ can distinguish between $\underline{div} \parallel_{Comm} \underline{stop}$ and $\underline{div}$. Thus in every consistent model $\mathcal{M}$ the law

$$\underline{div} \parallel_{Comm} \underline{stop} = \underline{div}$$

holds. But in $\mathcal{T}$ this law is false due to (*). (An analogous argument applies for the Counter Model $\mathcal{C}$.)

Nevertheless we can state:

Corollary 13.6 The Trace Model $\mathcal{T}$ and the Counter Model $\mathcal{C}$ are weakly consistent.

Proof. By Theorem 8.4 we can choose for $\mathcal{T}$ the following logical structure $\Vdash_{\mathcal{T}}$:

$$P \Vdash_{\mathcal{T}} s \text{ iff } \exists Q\colon P \xRightarrow{s} Q.$$

The logical structure $\Vdash_{\mathcal{C}}$ for $\mathcal{C}$ is then clear from Proposition 7.1. //

14. Conclusion

Starting from a simple idea of process correctness we developed a specific form of denotational sematics 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 $\longrightarrow$ ($- - \rightarrow$) denote (weak) homomorphisms.

Diagram 1

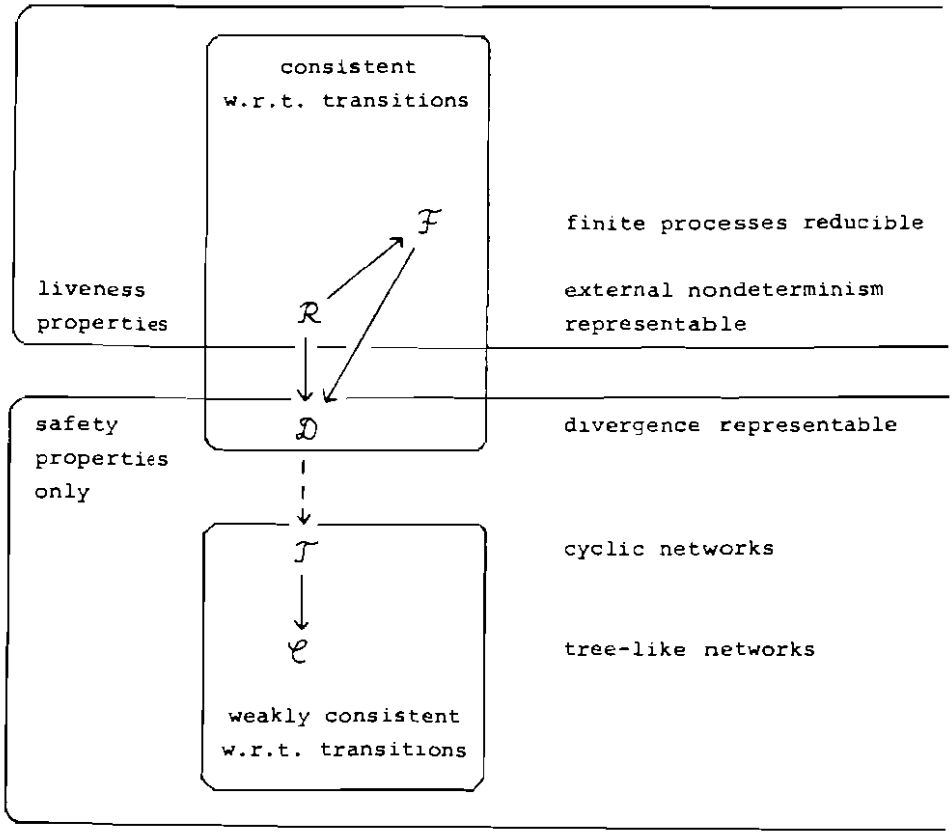


Diagram 1 explains the purposes and applications for which these models are best suited. For example, if we wish to reason about safety in divergence free cyclic networks of processes, we don't need the complex Failure Model $\mathcal{F}$; it suffices to choose the Trace Model $\mathcal{T}$. Also the models can be combined to new ones. For example, for reasoning about liveness properties in acyclic networks a simplified Readiness Model $\mathcal{R}-\mathcal{C}$ with communication counters instead of traces would do.

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 all the consequences of 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 successor relation $\longrightarrow$ is not image finite any more. Fortunately, such a change does not invalidate our continuity results in Section 5, but the extensibility condition for process specifications in Section 8 would require a clause ensuring bounded nondeterminism.

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 [44].

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 [10,20,31,45]. An advantage of starting from one of the models $\mathcal{C}$ to $\mathcal{F}$ would be that the question of completeness of the resulting proof system could be answered more transparantly [1,2,26,42].

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 conventional programming constructs and specifications expressed as predicates [15]. This style is expected to support a systematic development of concurrent programs from their specifications.

Acknowledgements

Preliminary versions of this paper were presented at informal seminars in Oxford, Leeds, Altenahr, Bad Honnef, Edinburgh and Yorktown Heights, at a meeting of the IFIP Working Group 2.2 in Venice, and at the ICALP '83 conference in Barcelona [35]. Comments, criticism and suggestions at these meetings helped very much to bring the paper into its present form. In particular, we wish to thank M. Broy, C. Crasemann, L. Czaja, M. Hennessy, G. Jones and A.W. Roscoe. The first author was moreover supported by the German Research Council (DFG) under grant No. La 426/3-1, 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] K.R. Apt, N. Francez, W.P. de Roever, A proof system for communication sequential processes, ACM TOPLAS 2 (1980) 359-385.

[2] K.R. Apt, Formal justification of a proof system for communicating sequential processes, J.ACM 30 (1983) 197-216.

[3] J.W. de Bakker, Mathematical theory of program correctness (Prentice Hall, London, 1980).

[4] J.W. de Bakker, J.I. Zucker, Processes and the denotational semantics of concurrency, Information and Control 54 (1982) 70-120.

[5] J.A. Bergstra, J.W. Klop, An abstraction mechanism for process algebras, Report IW 231/83, Mathematisch Centrum, Amsterdam (1983).

[6] J.D. Brock, W.B. Ackermann, Scenarios: a model for nondeterminate computations, in: J. Diaz, I. Ramos (Eds.), Formalisation of Programming Concepts, Lecture Notes in Computer Science 107 (Springer, Berlin-Heidelberg-New York, 1981) 252-267.

[7] S.D. Brookes, A model for communicating sequential processes, D.Phil. Thesis, Oxford Univ. (1983).

[8] S.D. Brookes, On the relationship of CCS and CSP, in: J. Diaz (Ed.), Proc. 10th Coll. Automata, Languages and Programming, Lecture Notes in Computer Science 154 (Springer, Berlin-Heidelberg-New-York, 1983) 83-96.

[9] M. Broy, Fixed point theory for communication and concurrency, in: D.Bjørner (Ed.), Formal Description of Programming Concepts II (North Holland, Amsterdam, 1983) 125-146.

[10] Zhou Chaochen, C.A.R. Hoare, Partial correctness of communicating processes, in: Proc. 2nd International Conference on Distributed Computing Systems, Paris (1981).

[11] N. Francez, C.A.R. Hoare, D.J. Lehmann, W.P.de Roever, Semantics of nondeterminism, concurrency and communication, JCSS 19 (1979) 290-308.

[12] N. Francez, D. Lehmann, A. Pnueli, A linear history semantics for languages for distributed programming, in: Proc. 21st IEEE Symp. on Foundations of Computer Science, Syracuse, N.Y. (1980).

[13] J.A. Goguen, J.W. Thatcher, E.G. Wagner, J.B. Wright, Initial algebra semantics and continuous algebras, J.ACM 24 (1977) 68-95.

[14] I. Guessarian, Algebraic semantics, Lecture Notes in Computer Science 99 (Springer, Berlin-Heidelberg-New-York, 1981).

[15] E.C.R. Hehner, Predicative programming, Part I and II, Comm. ACM 27 (1984) 134-151.

[16] E.C.R. Hehner, C.A.R. Hoare, A more complete model of communicating processes, TCS 26 (1983) 105-120.

[17] M. Hennessy, R. Milner, On observing nondeterminism and concurrency, in: J.W. de Bakker, J. van Leeuwen (Eds.) Proc. 7th Coll. Automata, Languages and Programming, Lecture Notes in Computer Science 85 (Springer, Berlin-Heidelberg-New York, 1980) 299-309.

[18] C.A.R. Hoare, Communicating sequential processes, Comm. ACM 21 (1978) 666-677.

[19] C.A.R. Hoare, A model for communicating sequential processes, in: R. M. Mc Keag, A.M. Mc Naghton (Eds.), On the Construction of Programs (Cambrigde University Press, 1980) 229-243.

[20] C.A.R. Hoare, A calculus of total correctness for communicating processes, Science of Computer Programming 1 (1981) 49-72.

[21] C.A.R. Hoare, Specifications, programs and implementations, Tech. Monograph PRG-29, Oxford Univ, Progr. Research Group, (1982).

[22] C.A.R. Hoare, S.D. Brookes, A.W. Roscoe, A theory of communicating sequential processes, Tech. Monograph PRG-16, Oxford Univ., Progr. Research Group, Oxford (1981); to appear in J.ACM.

[23] Ph. Jorrand, Specification of communicating processes and process implementation correctness, in: M. Dezani-Ciancaglini, U. Montanari (Eds.), Proc. 5th Intern. Symp. on Programming, Lecture Notes in Computer Science 137 (Springer, Berlin-Heidelberg-New York, 1983) 242-256.

[24] R. Keller, Formal verification of parallel programs, Comm. ACM 19 (1976) 371-384.

[25] J.R. Kennaway, C.A.R. Hoare, A theory of nondeterminism, in: J.W. de Bakker, J. van Leeuwen (Eds.) Proc. 7th Coll. Automata, Languages and Programming, Lecture Notes in Computer Science 85 (Springer, Berlin-Heidelberg-New York, 1980) 338-350.

[26] G.M. Levin, D. Gries, A proof technique for communicating sequential processes, Acta Inform. 15 (1981) 281-302.

[27] R. Milner, A calculus of communicating systems, Lecture Notes in Computer Science 92 (Springer, Berlin-Heidelberg-New-York, 1980).

[28] R. Milner, A modal characterisation of observable machine-behaviour, in: E. Astesiano, C. Böhm (Eds.), Proc. 6th Coll. Trees in Algebra and Programming, Lecture Notes in Computer Science 112 (Springer, Berlin-Heidelberg-New York, 1981) 25-34.

[29] R. Milner, Four combinators for concurrency, in: Proc. 1st ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, Ottawa (1982).

[30] J. Misra, K.M. Chandy, Proofs of networks of processes, IEEE Transactions on Software Engineering 7 (1981) 417-426.

[31] J. Misra, K.M. Chandy, T. Smith, Proving safety and liveness of communicating processes with examples, in: Proc. 1st ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, Ottawa (1982) 201-208.

[32] R. de Nicola, A complete set of axioms for a theory of communicating sequential processes, in: Proc. Intern. Conf. on Foundations of Computation Theory, Sweden (1983); to appear in Lecture Notes of Computer Science.

[33] R. de Nicola, M. Hennessy, Testing equivalences for processes, in: J. Diaz (Ed.), Proc. 10th Coll. Automata, Languages and Programming, Lecture Notes in Computer Science 154 (Springer, Berlin-Heidelberg-New York, 1983) 548-560.

[34] OCCAM, The OCCAM programming manual (INMOS, Bristol, 1982).

[35] E.-R. Olderog, C.A.R. Hoare, Specification-oriented semantics for communicating processes (preliminary version), in: J. Diaz(Ed.), Proc. 10th Coll. Automata, Languages and Programming, Lecture Notes in Computer Science 154 (Springer, Berlin-Heidelberg-New York, 1983) 561-572.

[36] S. Owicki, L. Lamport, Proving liveness properties of concurrent programs, ACM TOPLAS 4 (1982) 455-495.

[37] G.D. Plotkin, An operational semantics for CSP, in: D. Bjørner (Ed.), Formal Description of Programming Concepts II (North Holland, Amsterdam, 1983) 199-223.

[38] R. Reinecke, Networks of communicating processes: a functional implementation, Manuscript, Dept. of Comp. Sci., Univ. of Kaiserslautern (1983).

[39] A.W. Roscoe, A mathematical theory of communicating processes, D. Phil. Thesis, Oxford Univ. (1982).

[40] W.C. Rounds, S.D. Brookes, Possible futures, acceptances, refusals and communicating processes, in: Proc. 22nd IEEE Symp. on Foundations of Computer Science, Nashville, Tennessee (1981).

[41] M.B. Smyth, Power domains, JCSS 16 (1978) 23-26.

[42] N. Soundararajan, O.-J. Dahl, Partial correctness semantics of communicating sequential processes, Research Rep. No. 66, Inst. of Informatics, Univ. of Oslo (1982).

[43] J.E. Stoy, Denotational semantics: the Scott-Strachey approach to programming language theory (MIT Press, Cambridge, Mass., 1977).

[44] G. Winskel, Events in computation, Ph. D. Thesis, Dept. of Comp. Sci., Univ. of Edinburgh (1980).

[45] J. Zwiers, A.de Bruin, W.P. de Roever, A formal proof system for dynamically changing networks (Extended Abstract), in: D. Kozen, E.M. Clarke (Eds.), Proc. Logics of Programs, CMU, Pittsburgh (1983); to appear in Lecture Notes of Computer Science.