Impossible futures and determinism

Department of Mathematics and Computing Science

Impossible Futures and Determinism

by

M. Voorhoeve and. S. Mauw

00/14

ISSN 0926-4515

Computing Science Reports 00/14
Eindhoven, September 2000

Impossible Futures and Determinism

Marc Voorhoeve (wsinmarc@win.tue.nl),
Sjouke Mauw (sjouke@win.tue.nl)

Eindhoven University of Technology
PB 513, 5600 MB Eindhoven, the Netherlands

Abstract

The paper introduces a class $\preceq_n$ of process preorders that are related to contrasimulation equivalence. They are characterized by the constraints that they preserve. The preorder $\preceq_2$ (impossible futures) measures the "degree of determinism" and can be considered as the least discriminating preorder that can be used for the verification of communication protocols. If $p \preceq_2 q$ and q is deterministic, then p is deterministic too and the two are branching bisimilar. We present a system for (in)equational reasoning with the preorder $\preceq_2$ and indicate possible applications.

Keywords: Concurrency, Verification, Branching-Linear Spectrum

1 Introduction

There exist "pure" and "pragmatic" approaches to the specification and verification of concurrent systems. Purists regard a specification as a logic formula, a conjunction of *requirements*. *Safety* requirements state that the system is *not allowed* to exhibit some unwanted behavior, like S: "the system will not do b unless it has done a first". *Liveness* requirements state that the system is *guaranteed* to exhibit some desired behavior, like L: "the system will eventually do b". Verification consists of proving the specification formula for a given system p (the implementation) by proving that p satisfies each requirement R (notation $p \models R$).

If a denotes accepting a packet for transmission and b returning a correctly transmitted packet, L and S are requirements specifying a communication protocol. In Figure 1, two implementations are depicted. The system G starts with an a, then performs a sequence u of internal steps, after which a crucial nondeterministic internal transmission step t takes place. If the transmission succeeds, another sequence v of internal steps leads to b. If it fails, t will occur again after another sequence w of internal steps (e.g. after a time out) The system B has a third option: apart from successful or failed transmission, the system can reach a state where it will forever do w followed by t. Clearly, this is unwanted, whereas G is the best we can hope for if transmission is unreliable.

Nonetheless, both B and G satisfy S and neither satisfies L. It is possible that the transmission step in G fails over and over again. However, there exists a requirement L', slightly weaker than L such that $G \models L'$, but $B \not\models L'$. Such an L' is: "If the system has not yet done b, it is in a state

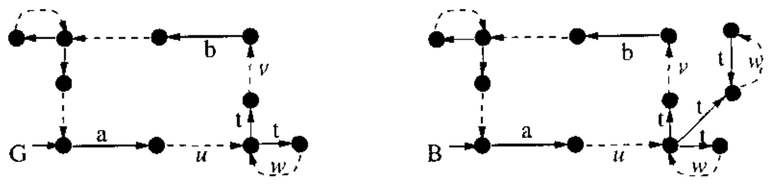

Figure 1: Good and bad communication protocols

where it can eventually do b". A similar requirement appears in [1]. We may deduce that $G \models L$ from a *fairness* assumption stating that events that always possible in the future will eventually occur. We will not invoke fairness, but rather use L' as our liveness requirement.

Although the purist approach is conceptually nice, the pragmatic school criticizes it on the following grounds. A specification is in most cases produced in close cooperation by software engineers and the (future) users of the system. The formal specification is created by the engineers and must be validated by the users. A specification by logic is hard to understand and validate by the users. It may be underspecific (allowing unwanted behaviors), overspecific (disallowing wanted behaviors) or both; detecting such errors requires hard work and considerable skill.

Pragmatists advocate specification-by-example. A (simple) system q serves as specification and a less simple system p (the implementation) is constructed and proved to satisfy $p \preceq q$ for some preorder $\preceq$. Even if the formalism is unknown by the users, an executable model can be built from the specification and experimented with until the users are convinced that it correctly describes the behavior they have in mind.

We do not want to pass any judgement in this debate. Most likely, combining both approaches will be most successful in practice. This means that we need a clear understanding of the classes of requirements that are and are not preserved by preorders.

There is a correspondence between preorders $\preceq$ and classes of requirements $\mathcal{C}(\preceq)$: for processes p, q we have $p \preceq q$ iff $q \models R \Rightarrow p \models R$ for each $R \in \mathcal{C}(\preceq)$. If a $\mathcal{C}(\preceq)$ is closed under negation (i.e. if $\neg R \in \mathcal{C}(\preceq)$ iff $R \in \mathcal{C}(\preceq)$), the preorder will be symmetric (i.e. an equivalence relation). If α, β are preorders and $\mathcal{C}(\alpha) \subseteq \mathcal{C}(\beta)$ then $p\ \beta\ q$ implies $p\ \alpha\ q$, so $\beta \subseteq \alpha$. We say that α is weaker than β.

Many preorders from literature (c.f. [8]) are based upon some notion of *finitary testing*. Since there is no testing scenario that can discriminate G and B in Figure 1, such preorders, like failures (c.f. [6]) or ready simulation (c.f. [5]), satisfy $B \preceq G$: a good specification can have a bad implementation.

The preorders not suffering from this drawback lie between contrasimilarity and branching bisimilarity (c.f. [8]) in Figure 2. These preorders are all based upon some notion of "global" or "fair" testing and are equivalent to strong bisimilarity if internal steps are absent. However, prescribing the behavior of a system up to (more or less) bisimilarity will often be overspecific. Therefore it is worthwhile to look for weaker preorders that still discriminate G and B in Figure 1.

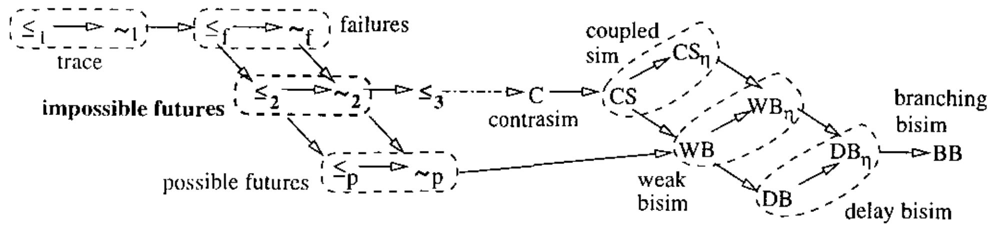

Figure 2: Spectrum from $\preceq_2$ to branching bisimilarity

In this paper we introduce a family of preorders $\preceq_n$ with $n \geq 0$, where $\preceq_{n+1} \subseteq \preceq_n$, $\preceq_1$ is trace inclusion and $\bigcap_{n\geq 0} \preceq_n$ is (almost) contrasimilarity. These preorders are characterized by classes $\mathcal{C}(\preceq_n)$ of requirements, defined in a HML-like (c.f. [10]) modal language. The preorders are precongruences for CCS-style operators and for $n > 1$ discriminate B and G in Figure 1. In Figure 2, they are listed with related other preorders. Inclusion is indicated by arrows there: $p \dashrightarrow q$ means that $q \subseteq p$. In the figure, the boldface preorders are the ones introduced here. The others are mentioned in [8]. Contrasimulation is introduced there, although in a very general setting.

The preorder $\preceq_2$ is investigated further. If $p \preceq_2 q$, then p and q have the same traces, but the moments of choice may differ; p may *delay* choices (c.f. [2]) made by q, thus being "more deterministic" than q. An important property is that if $p \preceq_2 q$ and q is deterministic, then p is weakly (and by Theorem 3.1 in [9] also branching) bisimilar to q. So, for deterministic specifications all preorders in the spectrum between $\preceq_2$ and branching bisimilarity (c.f. Figure 2) collapse.

We present a deductive system for $\preceq_2$ and give a toy example that illustrates its use for specification and verification.

Acknowledgements

We thank an anonymous referee of an earlier version and our colleagues Jos Baeten, Ruurd Kuiper and Bas Luttik. A special acknowledgement is deserved by Rob van Glabbeek who is the godfather of the ideas which led to this paper.

2 Basic notions

In this section we fix some notation. If $k \geq 0$ and $X_1, \ldots, X_k$ are sets, $X_1 \times \ldots \times X_k$ is the set of k-tuples $(x_1, \ldots, x_k)$ with $x_i \in X_i$ for all i with $1 \leq i \leq k$. The empty product (with $k = 0$) is the singleton set $\{\epsilon\}$, so ϵ denotes the 0-tuple. The set X^k is the k-fold product $X \times \ldots \times X$ and $X^* = \bigcup_{k\geq 0} X^k$. Juxtaposition combines tuples, e.g $(x, y)(u, v, w) = (x, y, u, v, w)$. We identify X^1 and X.

Binary relations are sets of 2-tuples (pairs). We write $x\ R\ y$ if the pair (x, y) is an element of the relation R and $x\ R$ iff $\exists y :: x\ R\ y$. The operator $\_ \circ \_$ denotes relation composition and $\_^{-1}$ denotes relation inversion.

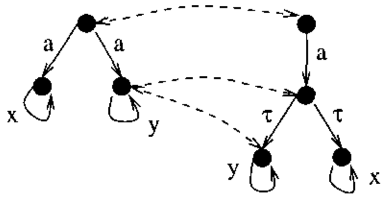

Figure 3: A contrasimulation

A *preorder* is a reflexive and transitive relation. A preorder $\preceq$ on a set X is a *precongruence* w.r.t. a function $f : X^k \to X$ iff $x_i \preceq y_i$ for all i with $1 \leq i \leq k$ implies that $f(x_1, \ldots, x_k) \preceq f(y_1, \ldots, y_k)$.

Throughout this paper, A will be a set of (visible) actions, τ (with $\tau \notin A$) is the invisible or internal action, $A_\tau = A \cup \{\tau\}$, $\mathcal{X}$ is a set of states or *processes* and for $a \in A_\tau$, $\xrightarrow{a} \subseteq \mathcal{X} \times \mathcal{X}$ are the transition relations. The set $\mathcal{X}$ is assumed to be large enough to contain all processes that we will consider. A state x is called *unstable* iff $x \xrightarrow{\tau}$ and *stable* otherwise. If $x \xrightarrow{a} x'$, this is interpreted as the occurrence of an action a in a state x, resulting in a (possibly different) state x'. Note that $\mathcal{X}$ is supposed to contain all processes; instead of comparing different process spaces, we take the union of their states and transition relations and compare the processes in it.

We define the relations $\overset{\sigma}{\Longrightarrow} \subseteq \mathcal{X} \times \mathcal{X}$ for $\sigma \in A^*$ as the least (w.r.t. inclusion) relation satisfying $x \overset{\epsilon}{\Longrightarrow} x$, $\dfrac{x \xrightarrow{a} x', x' \overset{\sigma}{\Longrightarrow} x'', a \in A}{x \overset{a\sigma}{\Longrightarrow} x''}$, $\dfrac{x \xrightarrow{\tau} x', x' \overset{\sigma}{\Longrightarrow} x''}{x \overset{\sigma}{\Longrightarrow} x''}$. So $x \overset{\epsilon}{\Longrightarrow} x'$ iff x' can be reached from x by executing 0 or more internal steps. The subrelation $\overset{\epsilon+}{\Longrightarrow}$ requires at least one internal step, so $x \overset{\epsilon+}{\Longrightarrow} x'$ iff there exists an x'' such that $x \overset{\epsilon}{\Longrightarrow} x''$ and $x'' \xrightarrow{\tau} x'$. For $\sigma \neq \epsilon$ we set $\overset{\sigma+}{\Longrightarrow} = \overset{\sigma}{\Longrightarrow}$. A *trace* of a process x is a sequence $\sigma \in A^*$ such that $x \overset{\sigma}{\Longrightarrow}$.

A process x has *finite nondeterminism* iff for each $\sigma \in A^*$ the set $\{x' \mid x \overset{\sigma}{\Longrightarrow} x'\}$ is finite. The set $\mathcal{X}^F$ is the set of all processes with this property.

A *contrasimulation* (c.f. [8]) is a relation $R \subseteq (\mathcal{X} \times \mathcal{X})$ such that for all $\sigma \in A^*$, $(R^{-1} \circ \overset{\sigma}{\Longrightarrow}) \subseteq (\overset{\sigma}{\Longrightarrow} \circ R)$. This means that if $p \; R \; q$ and $p \overset{\sigma}{\Longrightarrow} p'$, there exists a q' such that $q \overset{\sigma}{\Longrightarrow} q'$ and $q' \; R \; p'$. Note the inversion (the "contra" of contrasimulation)! In Figure 3, a contrasimulation is depicted.

A coupled simulation is a contrasimulation R satisfying $\{x \mid x \; R\} = \{y \mid R \; y\}$. A *weak bisimulation* is a symmetric contrasimulation. Two states x, y of $\mathcal{X}$ are *weakly bisimilar / coupled similar / contrasimilar* iff there exists a weak bisimulation / coupled simulation / contrasimulation R such that $x \; R \; y$ and $y \; R \; x$. The relations are denoted respectively as $\sim_{WB}$, $\sim_{CS}$ and $\sim_C$.

Our requirement language $\mathcal{L}$ is composed from the constant $\top$, the unary operators $\neg$ and $\Box_S$ (with $S \subseteq A^*$) and the binary operator $\wedge$.

A requirement L inductively defines a set $[\![L]\!] \subseteq \mathcal{X}$ of processes as follows.
$[\![\top]\!] = \mathcal{X}$, $[\![\neg M]\!] = \mathcal{X} \setminus [\![M]\!]$, $[\![M \wedge N]\!] = [\![M]\!] \cap [\![N]\!]$ and $[\![\Box_S M]\!] = \{p \mid \forall p' : (\exists \sigma \in S :: p \overset{\sigma}{\Longrightarrow} p') : p' \in [\![M]\!]\}$.

We write $p \models L$ ("p satisfies L") iff $p \in [\![L]\!]$.

Abbreviations are $\bot = \neg\top$, $L \vee M = \neg(\neg L \wedge \neg M)$, $L \Rightarrow M = \neg L \vee M$ and $\Diamond_S L = \neg\Box_S\neg L$. We write $\Box_\sigma$, $\Diamond_\sigma$ instead of $\Box_{\{\sigma\}}$, $\Diamond_{\{\sigma\}}$

If $p, q \in \mathcal{X}^F$, then $p \sim_{WB} q$ iff $\forall L \in \mathcal{L} : q \models L \Leftrightarrow p \models L$. We call process p *deterministic* iff $p \models (\Diamond_{\rho\sigma}\top \Rightarrow \Box_\rho\Diamond_\sigma\top$ for all $\rho, \sigma \in A^*$. This means that if a behavior $\rho\sigma$ is possible, the behavior σ will be possible after having observed ρ.

3 A family of preorders

We shall give a relational definition for the preorders $\preceq_n$ and determine sets $\mathcal{C}(\preceq_n)$ of requirements for them. A similar characterization will be given for contrasimilarity. We will use auxiliary relations $\ll_n$. Informally, $p \ll_n q$ iff every computation in p leading to a state p' can be matched by a similar computation in q leading to q' in such a way that $q' \ll_{n-1} p'$. Again, note the inversion.

Definition 1 *Let p, q be processes. Then $p \ll_0 q$ and for each $n \geq 0$,*
$p \ll_{n+1} q \Leftrightarrow \forall\sigma, p' : p \overset{\sigma}{\Longrightarrow} p' : (\exists q' : q \overset{\sigma}{\Longrightarrow} q' : q' \ll_n p')$.

Note that $p \ll_1 q$ iff $\forall\sigma : p \overset{\sigma}{\Longrightarrow} : q \overset{\sigma}{\Longrightarrow}$, which means that every trace of p is also a trace of q. The following propositions connect the relations $\ll_n$ to one another and to contrasimilarity.

Proposition 1 *For all $n \geq 0$, $\sim_C \subseteq \ll_{n+1} \subseteq \ll_n$.*

Proof: Let $p, q \in \mathcal{X}$ and $n \geq 0$. By induction on n, we prove that (a) the existence of a contrasimulation R such that $p\ R\ q$ implies that $p \ll_n q$ and that (b) $p \ll_{n+1} q$ implies that $p \ll_n q$. From (a) follows that $\sim_C \subseteq \ll_{n+1}$ for any $n \geq 0$. The base case $n = 0$ is immediate for both. So let $n > 0$.

Let R be a contrasimulation such that $p\ R\ q$. We want to prove that $p \ll_n q$. So suppose $p \overset{\sigma}{\Longrightarrow} p'$. Since R is a contrasimulation, there exists a q' such that $q \overset{\sigma}{\Longrightarrow} q'$ and $q'\ R\ p'$. By IH, $q' \ll_{n-1} p'$. So indeed $p \ll_n q$, proving (a).

Suppose $p \ll_{n+1} q$. We want to prove that $p \ll nq$. So let $p \overset{\sigma}{\Longrightarrow} p'$. There must exist a q' such that $q \overset{\sigma}{\Longrightarrow} q'$ and $q' \ll_n p'$. By IH, $q' \ll_{n-1} p'$, so indeed $p \ll_n q$, proving (b). □

Proposition 2 *If $p, q \in \mathcal{X}^F$, then $p \sim_C q \Leftrightarrow \forall n > 0 : p \ll_n q \wedge q \ll_n p$.*

Proof: One side of the implication follows from the previous proposition. For the other side, we show that $R = \bigcap_n \ll_n$ is a contrasimulation. Let $p\ R\ q$ and $p \overset{\sigma}{\Longrightarrow} p'$. Then for any n, $p \ll_{n+1} q$, so there exists a q' such that $q \overset{\sigma}{\Longrightarrow} q' \wedge q' \ll_n p'$. So for any n there exists a q' such that $q \overset{\sigma}{\Longrightarrow} q' \wedge q' \ll_n p'$. Since $q \in \mathcal{X}^F$, there are but finitely many q' such that $q \overset{\sigma}{\Longrightarrow} q'$, so there is a q' such that $q \overset{\sigma}{\Longrightarrow} q'$ and $q' \ll_n p'$ for all n, so $q' \; R \; p'$. □

We now will define sets C_n^+ of requirements.

Definition 2 *For $n \in \mathbb{N}$ we define subsets C_n^+ of $\mathcal{L}$ as the smallest sets satisfying*

$$\frac{}{\top \in C_0^+}, \quad \frac{L, M \in C_n^+}{L \wedge M \in C_n^+, L \vee M \in C_n^+}, \quad \frac{L \in C_n^+, S \in \mathcal{P}(A^*)}{\Box_S \neg L \in C_{n+1}^+}.$$

Many requirements and theoretical properties can be stated within C_1^+ and C_2^+.. The requirement S and L' in the introduction are respectively $\Box_{\{\sigma b | \sigma \in (A \setminus \{a\})^*\}} \bot$, which is in C_1^+ and $\Box_{(A \setminus \{b\})^*} \Diamond_{\{\rho b | \rho \in A^*\}} \top$, which is in C_2^+. Also the property of being σ, ρ-deterministic: $\Box_{\rho\sigma} \bot \vee \Box_\rho \Diamond_\sigma \top$ is in C_2^+.

Theorem 1 *Let $p, q \in \mathcal{X}^F$, $n \in \mathbb{N}$. Then $p \ll_n q \Leftrightarrow \forall L \in C_n^+ : q \models L : p \models L$.*

Proof: Suppose $p \ll_n q$ and let $L \in C_n^+$ such that $q \models L$. We use structure induction on L to prove that $p \models L$. We may assume $n > 0$. The base case $L \in \{\top, \bot\}$ is immediate. If $L = M \wedge N$ then $q \models M \wedge q \models N$, so by IH $p \models M \wedge p \models N$, thus $p \models L$. The case $L = M \vee N$ is similar. The remaining case is $L = \Box_S \neg M$, with $M \in C_{n-1}^+$. Assume $p \not\models L$, so $p \models \Diamond_S M$. So there exists a $\sigma \in S$ and a p' such that $p \overset{\sigma}{\Longrightarrow} p'$ and $p' \models M$. From the definition of $\ll_n$, there exists a q' such that $q \overset{\sigma}{\Longrightarrow} q'$ and $q' \ll_{n-1} p'$, so by IH (on the structure!), we conclude that $q' \models M$ and thus $q \not\models L$, which contradicts our assumption. So $p \models L$ in all cases.

Conversely suppose $p \not\ll_n q$. By induction on n we prove that there exists an $L \in C_n^+$ such that $q \models L$ and $p \not\models L$. The case $n = 0$ is trivial, so let $n > 0$. Since $p \not\ll_n q$, there exists a p' with $p \overset{\sigma}{\Longrightarrow} p'$ such that for all q_i' with $q \overset{\sigma}{\Longrightarrow} q_i'$, $q_i' \not\ll_{n-1} p'$. Since $q \in \mathcal{X}^F$, there are but finitely many such q_i'. By IH, for each such q_i' there is a L_i such that $p' \models L_i$ and $q_i' \not\models L_i$. So $q \models \Box_\sigma \bigwedge_i \neg L_i$ and $p \not\models \Box_\sigma \bigwedge_i \neg L_i$. □

We now define some derived relations and requirements.

Definition 3 *For each $n \geq 0$ we define the following relations and requirement sets.*

$$\begin{array}{lll} \gg_n \; = \; \ll_n^{-1} & \preceq_{n+1} \; = \; \ll_{n+1} \cap \gg_n & \sim_n = \ll_n \; \cap \; \gg_n \\ C_n^- = \{\neg L \mid L \in C_n^+\} & \mathcal{C}(\preceq_{n+1}) = C_{n+1}^+ \cup C_n^- & C_\omega = \bigcup_{n>0} \mathcal{C}(\preceq_n) \end{array}$$

Simple set theory yields a.o. the following results (that go without proof).

Proposition 3 *Let $n \geq 0$. Then $\sim_C \subseteq \preceq_{n+1} \subseteq \preceq_n$.*
If $p, q \in \mathcal{X}^F$, then $p \preceq_n q \Leftrightarrow \forall L \in \mathcal{C}(\preceq_n) : q \models L : p \models L$
and also $p \sim_C q \Leftrightarrow \forall L \in C_\omega : q \models L : p \models L$.

The formulae in C_ω are characterized by an alternation of $\Box$ and $\Diamond$ operators. Indeed, C_ω cannot contain the formula $N = \Diamond_a(\Diamond_x\top \wedge \Diamond_y\top)$, which discriminates the processes in Figure 3. Instead, C_ω contains e.g. $\Diamond_a\Box_\epsilon(\Diamond_x\top \wedge \Diamond_y\top)$.

The preorder $\preceq_{n+1}$ forms a lattice on the $\sim_n$-equivalent processes. The preorder $\bigcup_{n>0} \preceq_n$ defines a lattice on $\mathcal{X}$ as a whole. The preorder $\preceq_2$ has been named "impossible futures preorder" after [12]. Impossible futures of a p are pairs $(\sigma, F) \in A^* \times \mathcal{P}(A^*)$ such that $p \models \Diamond_\sigma\Box_F\bot$, so after having observed σ, it is possible that no behavior from F can be observed anymore. Every impossible future of p is an impossible future of q iff $p \ll_2 q$. We now prove the determinism property mentioned in the introduction, which is largely due to [7].

Theorem 2 *Let $p \ll_2 q$ and q deterministic. Then $p \sim_{WB} q$.*

Proof: Determinism is a C_2^+ requirement, so by Theorem 1, p must be deterministic as well. We first show that p and q are trace equivalent, i.e. that $\forall\sigma \in A^* :: p \models \Diamond_\sigma\top \Leftrightarrow q \models \Diamond_\sigma\top$. Note that $\|\bot\| = \|\Diamond_\epsilon\bot\|$. So suppose $p \models \Diamond_\sigma\top$, so $p \not\models \Box_\sigma\bot$, so $p \not\models \Box_\sigma\Diamond_\epsilon\bot$, so since $p \ll_2 q$, $q \not\models \Box_\sigma\Diamond_\epsilon\bot$, so $q \models \Diamond_\sigma\top$. Suppose $q \models \Diamond_\sigma\top$, so since q is deterministic, $q \models \Box_\epsilon\Diamond_\sigma\top$, so since $p \ll_2 q$, $p \models \Box_\epsilon\Diamond_\sigma\top$, so $p \models \Diamond_\sigma\top$.

We now shall prove that if p, q are deterministic and trace equivalent then $p \sim_{WB} q$. Let R be defined by $x\ R\ y \Leftrightarrow \exists\sigma : p \overset{\sigma}{\Longrightarrow} x \wedge q \overset{\sigma}{\Longrightarrow} y$. We will prove that $R \cup R^{-1}$ is a (symmetric!) contrasimulation. Suppose $x\ R\ y$ and $x \overset{\sigma}{\Longrightarrow} x'$. There must exist a ρ such that $p \overset{\rho}{\Longrightarrow} x$ or $q \overset{\rho}{\Longrightarrow} x$. Since we have symmetry between p and q we may assume wlog the former. So we have $p \overset{\rho\sigma}{\Longrightarrow} x'$. Since $p \preceq_1 q$, we have $q \models \Diamond_{\rho\sigma}\top$, so by the determinism of q, $q \models \Box_\rho\Diamond_\sigma\top$ and since $q \overset{\rho}{\Longrightarrow} y$, we must have $y \models \Diamond_\sigma\top$, so there is a y' such that $y \overset{\sigma}{\Longrightarrow} y'$ and thus $y'\ R\ x'$. $\Box$

4 Operators

In this section we introduce some ACP-like (cf. [4]) operators and show that our preorders are precongruences w.r.t. them.

We presuppose a ternary communication relation $\gamma \subseteq A^3$. By imposing additional constraints upon γ, the standard ACP merge is obtained. The process δ denotes inaction. In Table 1 we give SOS rules for the following operators: choice $(\_ + \_)$, merge $(\_\|\_)$, action prefix ($a\_$, with $a \in A_\tau$), encapsulation ($\partial_H(\_)$, with $H \subseteq A$) and renaming ($\rho_r(\_)$, with $r \in A \to (A_\tau)$). In Figure 4, processes p, q are shown with some processes derived from them by these operators (assuming that $(a, b, x) \in \gamma \Leftrightarrow x = c$).

We introduce a definition and a lemma about the traces of merged processes.

Definition 4 *The trace weave operator $\_|\_ \in A^* \times A^* \to \mathcal{P}(A^*)$ is inductively defined as follows:*
$\epsilon|\sigma = \sigma|\epsilon = \{\sigma\}$,
$a\rho|b\sigma = \{a\pi \mid \pi \in \rho|b\sigma\} \cup \{b\pi \mid \pi \in a\rho|\sigma\} \cup \{c\pi \mid (a, b, c) \in \gamma \wedge \pi \in \rho|\sigma\}$

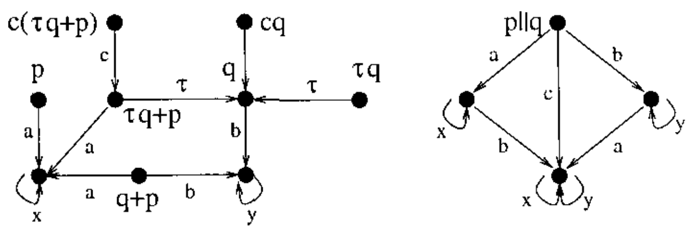

Figure 4: Derived processes

$$ap \xrightarrow{a} p \qquad \frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'\ ,\ q+p \xrightarrow{a} p'}$$

$$\frac{p \xrightarrow{a} p'}{p||q \xrightarrow{a} p'||q\ ,\ q||p \xrightarrow{a} q||p'} \qquad \frac{p \xrightarrow{a} p'\ ,\ q \xrightarrow{b} q'\ ,\ (a,b,c) \in \gamma}{p||q \xrightarrow{c} p'||q'}$$

$$\frac{p \xrightarrow{a} p'\ ,\ a \notin H}{\partial_H(p) \xrightarrow{a} \partial_H(p')} \qquad \frac{p \xrightarrow{a} p'}{\rho_r(p) \xrightarrow{r(a)} \rho_r(p')}$$

Table 1: SOS rules for simple operators

Lemma 1 $p||r \overset{\sigma}{\Longrightarrow} s \Leftrightarrow \exists p', r', \pi, \rho : s = p'||r' \wedge p \overset{\pi}{\Longrightarrow} p' \wedge r \overset{\rho}{\Longrightarrow} r' \wedge \sigma \in \pi|\rho$.

Proof: Let $p||r \overset{\sigma}{\Longrightarrow} s$, so there are $s_0 \dots s_n$ such that $p||r = s_0, s = s_n$ and $s_i \xrightarrow{a_i} s_{i+1}$ for $0 \leq i < n$ and $a_i \in A_\tau$ and $\phi(a_0, \dots, a_{n-1}) = \sigma$, where $\phi : A_\tau^* \to A^*$ strikes out τ's.
We use induction on n to show that this is equivalent to $\exists p', r', \pi, \rho :: s = p'||r' \wedge p \overset{\pi}{\Longrightarrow} p' \wedge r \overset{\rho}{\Longrightarrow} r' \wedge \sigma \in \pi|\rho$. If $n = 0$ then $s = p||r, \sigma = \epsilon$ and we set $p' = p, r' = r, \pi = \rho = \epsilon$. So let $n > 0$. The rules in Table 1 allow three possibilities for s_1, namely $p_1||r$ if $p \xrightarrow{a_0} p_1$, $p||r_1$ if $r \xrightarrow{a_0} r_1$ and $p_1||r_1$ if $p \xrightarrow{b_0} p_1, r \xrightarrow{c_0} r_1$ and $(a_0, b_0, c_0) \in \gamma$.
Assuming the first, and assuming $a_0 \neq \tau$, we must have that $\sigma = a_0\sigma_1$ and $p_1||r \overset{\sigma_1}{\Longrightarrow} s$, so the IH yields that this is equivalent to $\exists p', r', \pi, \rho :: s = p'||r' \wedge p_1 \overset{\pi}{\Longrightarrow} p' \wedge r \overset{\rho}{\Longrightarrow} r' \wedge \sigma_1 \in \pi|\rho$. Thus, $\exists p', r', \pi, \rho :: s = p'||r' \wedge p \overset{a_0\pi}{\Longrightarrow} \wedge r \overset{\rho}{\Longrightarrow} r' \wedge \sigma \in (a_0\pi)|\rho$. The other cases are similar.
□

We further restrict our preorders by a root condition.

Definition 5 *Let p and q be processes. Then $p \ll_n^r q$ iff $p \ll_n q$ and either $n = 1$ or $p \xrightarrow{\tau} \Rightarrow q \xrightarrow{\tau}$. For $n > 1$, we set $\preceq_n^r = \ll_n^r \cap (\ll_{n-1}^r)^{-1}$.*

We give a lemma about unstable processes. Its proof follows from the definition of $\ll_n$.

Lemma 2 *Let p, p' and q be processes such that $p \xrightarrow{\tau} p'$ and $p \ll_n q$ for some $n \geq 0$. Then also $p' \ll_n q$.*

Note that the processes in Figure 4 satisfy $\tau q \sim_n q$ and $q \ll_n \tau q + p$ for all $n > 1$, but $\tau q + p \not\sim_n q + p$ and $cq \not\ll_n c(\tau q + p)$. This shows that the preorders $\sim_n$, $\preceq_2$ and $\ll_n$ are not precongruences for every defined operator.

Theorem 3 *The $\preceq_n^r$ are precongruences for the operators defined in Table 1.*

Proof:

We first prove $p \ll_n q \Rightarrow p||r \ll_n q||r$ by induction on n. The case $n = 0$ is trivial, so let $n > 1$ and $p \ll_n q$. Suppose $p||r \overset{\sigma}{\Longrightarrow} s$. By Lemma 1, there exist p', r', ρ, π such that $p \overset{\pi}{\Longrightarrow} p', r \overset{\rho}{\Longrightarrow} r'$ and $\sigma \in \pi|\rho$. Since $p \ll_n q$, there exists a q' such that $q \overset{\pi}{\Longrightarrow} q'$ and $q' \ll n-1 p'$. By IH, $q'||r' \ll n-1 p'||r'$ and by Lemma 1 $q||r \overset{\sigma}{\Longrightarrow} q'||r'$. So $p||r \ll_n q||r$. From $p \xrightarrow{\tau} \Rightarrow q \xrightarrow{\tau}$ follows $p||r \xrightarrow{\tau} \Rightarrow q||r \xrightarrow{\tau}$, so $p \ll_n^r q \Rightarrow p||r \ll_n^r q||r$. From $q \ll_{n-1} p$ follows $q||r \ll_{n-1} p||r$. So $p \preceq_n^r q \Rightarrow p||r \preceq_n^r q||r$. By symmetry, $p \preceq_n^r q \Rightarrow r||p \preceq_n^r r||q$. So if $p \preceq_n^r q$ and $r \preceq_n^r s$, we have $p||r \preceq_n^r q||r \preceq_n^r q||s$, so the $\preceq_n^r$ are precongruences for the merge operator.

Next, we prove that $p \preceq_n q \Rightarrow \alpha p \preceq_n \alpha q$ for $\alpha \in A_\tau$ by a similar induction. Suppose $\alpha p \overset{\sigma}{\Longrightarrow} s$. Then either $\sigma = \epsilon, s = \alpha p$ or $\sigma = [\alpha\pi], s = p', p \overset{\pi}{\Longrightarrow} p'$. Here $[\alpha\pi] = \pi$ if $\alpha = \tau$ and $\alpha\pi$ otherwise. In the first case we know that $q \preceq_{n-1} p$, so by IH, $\alpha q \preceq_{n-1} \alpha p$, so there exists an r (namely αq) such that $\alpha q \overset{\sigma}{\Longrightarrow} r$ and $r \preceq_{n-1} s$. In the second case there exists a q' such that $q \overset{\pi}{\Longrightarrow} q'$ (thus $\alpha q \overset{\sigma}{\Longrightarrow} q'$) and $q' \preceq n-1 p'$. In either case the condition that $\alpha p \ll_n \alpha q$ is met. As in the merge case, this implies that $\alpha p \preceq_n \alpha q$. Also, $\alpha p \xrightarrow{\tau}$ implies $\alpha q \xrightarrow{\tau}$.

Finally, we prove that $p \ll_n^r q \Rightarrow p + r \ll_n^r q + r$. Suppose $p + r \overset{\sigma}{\Longrightarrow} s$. Then either $\sigma = \epsilon, s = p + r$ or $s = q', q \overset{\sigma+}{\Longrightarrow} q'$ or $s = p', p \overset{\sigma+}{\Longrightarrow} p'$. For the first two cases, the induction step is easy, so we assume the third case. Since $p \preceq_n^r q$ and $p \overset{\sigma+}{\Longrightarrow} p'$, there must be a q' such that $q \overset{\sigma}{\Longrightarrow} q'$ and $q' \ll_{n-1} p'$. If $q \neq q'$ or $\sigma \neq \epsilon$, we have $q + r \overset{\sigma}{\Longrightarrow} q'$, completing the induction. So suppose $\sigma = \epsilon$ and $q' = q$. Since $p \overset{\sigma+}{\Longrightarrow} p'$, we have that $p \xrightarrow{\tau}$ and since $p \preceq_n^r q$, there is a q'' such that $q \xrightarrow{\tau} q''$. By Lemma 2, we have $q + r \overset{\sigma}{\Longrightarrow} q''$ and $q'' \ll_{n-1} p'$, completing the induction in the last case. As above, this implies that $p \preceq_n^r q \Rightarrow p + r \preceq_n^r q + r$. By symmetry, $p \preceq_n^r q \Rightarrow r + p \preceq_n^r r + q$.

The remaining operators are similar to the merge case. For a given operator ϕ, we characterize the s, σ pairs such that $\phi(p) \overset{\sigma}{\Longrightarrow} s$, like we did in Lemma 1. In all cases, there must exist a ρ (depending upon ϕ and σ) and a p' such that $p \overset{\rho}{\Longrightarrow} p'$ and $s = \phi(p')$. From such a characterization, the proof by induction is straightforward. □

A special case of the renaming operator is the abstraction operator $\tau_H(\_)$ with $H \subseteq A$. We have $\tau_H = \rho_{F_H}$, where $F_H(a) = \tau$ if $a \in H$ and $F_H(a) = a$ otherwise.

With the above operators we can construct *finite* processes (i.e. with finite trace sets). For infinite processes we define a simple recursion operator.

For reasoning with infinite processes we introduce the projection operators. Table 2 contains the SOS rules for the process $[M]_i$, with I some (not necessarily finite) index set, $M \subseteq (I \times A \times I)$, $i \in I$, and $\pi_n(\_)$, with $n \in \mathbb{N}$. By the SOS rules, $[M]_i = \delta$ if M contains no triple (i, a, j), e.g.

$$\frac{(i,a,j) \in M}{[M]_i \xrightarrow{a} [M]_j} \qquad \frac{p \xrightarrow{a} p' \ , \ a \neq \tau \ , \ n > 0}{\pi_n(p) \xrightarrow{a} \pi_{n-1}(p')} \qquad \frac{p \xrightarrow{\tau} p'}{\pi_n(p) \xrightarrow{\tau} \pi_n(p')}$$

Table 2: SOS rules for recursion/projection

when $M = \emptyset$. Note that τ's are not allowed in the relation M. If this were allowed, Theorem 4 would no longer be valid. To construct infinite processes with silent steps, we first construct processes without them and use abstraction afterward.

We can now prove the following theorem, showing that the $\preceq_n$ are precongruences for projection and the approximation induction principle (AIP) for the preorders.

Theorem 4 *Let $m, n \geq 0$. The preorder $\preceq_n$ is a precongruence for π_m. For all $p, q \in \mathcal{X}^F$, we have $p \preceq_n q \Leftrightarrow \forall i \geq 0 : \pi_i(p) \preceq_n \pi_i(q)$*

Proof: The properties are proved by induction on n. The essential part of the proof is the characterization $\pi_m(p) \overset{\sigma}{\Longrightarrow} s \Leftrightarrow \ell(\sigma) \leq m \ \wedge \ \exists p' : p \overset{\sigma}{\Longrightarrow} p' \ \wedge \ \pi_{m-\ell(\sigma)}p' = s$, where the function ℓ gives the length of a trace. The finite nondeterminism property is needed to conclude that $\forall n \geq 0 : (\exists q' : q \overset{\sigma}{\Longrightarrow} q' : q' \ll_{n-1} p')$ implies that $\exists q' : q \overset{\sigma}{\Longrightarrow} q' : (\forall n \geq 0 : q' \ll_{n-1} p')$. □

5 Calculus

In this section we give an axiomatization of $\preceq_2^r$ for finite processes. Let A be a set of actions as before and V with $V \cap A = \emptyset$ a set of variables. We present a deductive system $\Delta_{A,V}$ for process terms with a relation $\leq$ that will axiomatize $\preceq_2^r$. In this section we will abbreviate $\preceq_2^r$ by $\preceq$.

Table 3 presents the axioms for finite process terms. The relation $=$ in the axioms is an abbreviation for $\leq \cap \geq$. The terms are built from δ, process variables $(x, y, z \in V)$, the silent action (τ), actions $(a \in A_\tau)$, the action prefix $(a\_)$ and choice $(\_ + \_)$ operator. Brackets are used to indicate the order in which the operators are applied. If omitted, action prefix binds stronger than choice.

The axioms in Table 3 do not contain the merge, encapsulation or renaming operators. However, there exist rewrite rules (e.g. an expansion theorem for the merge operator) that allow every term without variables containing these operators to be represented as a $\Delta_{A,V}$ term.

A1	$x + y = y + x$	IF	$x \leq \tau x$
A2	$(x + y) + z = x + (y + z)$	CS	$\tau(\tau x + y) = \tau x + y$
A3	$x + x = x$	C	$ax + ay = a(\tau x + \tau y)$
A6	$x + \delta = x$		

Table 3: Basic axioms

The A axioms define (strong) bisimilarity. The axiom CS stems from the axiomatization of coupled simulation (cf. [11]). With the axiom C and the axiom T2: $\tau x + x = \tau x$ it axiomatizes contrasimilarity for finite terms (cf. [8]). Also compare Figure 2. The axioms IF and C can be found in [6]. The axiom C connects stable nondeterminism (several possible outcomes from a visible step) to unstable nondeterminism (several possible outcomes from invisible step). Axiom IF states that the addition of a silent step makes the process less deterministic.

The deduction rules that we may use are RE(flexivity), i.e. $t \leq t$, TR(ansitivity), i.e. $t \leq u \wedge u \leq v \Rightarrow t \leq v$, IN(stantiation), i.e. $E(x) \leq F(x) \Rightarrow E(t) \leq F(t)$ and SU(bstitutivity), i.e. $t \leq u \Rightarrow E(t) \leq E(u)$, where t, u, v are arbitrary terms, x a variable and $E(x)$, $F(x)$ terms containing x. The term $E(t)$ denotes the term obtained from $E(x)$ by substituting all occurrences of x by t.

We show an example deduction, deriving (T2): $\tau x + x = \tau x$

true $\overset{\text{IF}}{\Rightarrow} x \leq \tau x \overset{\text{SU}}{\Rightarrow} (\tau x + x) \leq (\tau x + \tau x) \overset{\text{A3,IN}}{\Rightarrow} (\tau x + x) \leq \tau x$.

true $\overset{\text{IF,SU,A3}}{\Rightarrow} x \leq (\tau x + x) \overset{\text{SU}}{\Rightarrow} \tau x \leq \tau(\tau x + x) \overset{\text{CS,IN,SU}}{\Rightarrow} \tau x \leq (\tau x + x)$.

Another derivation (omitting IN, SU, TR and A2) yields $\tau x + y = \tau x + \tau(x + y)$:

$\tau x + y \overset{\text{T2}}{=} \tau x + x + y \leq^{\text{IF}} \tau x + \tau(x + y)$

$\tau x + \tau(x + y) \leq^{\text{IF}} \tau x + \tau(\tau x + y) \overset{\text{CS}}{=} \tau x + \tau x + y \overset{\text{A3}}{=} \tau x + y$.

A third derivation features the delay of a choice: $a(x + y) \leq^{\text{IF}} a(\tau x + \tau y) \overset{\text{C}}{=} ax + ay$.

Because of axioms A1, A2 and A6, we can use the notation $\sum_{i \in I} E_i$ for processes, provided that the set I is finite. Its meaning is $E_1 + \ldots + E_n$, where $E_1, \ldots E_n$ is some ordering of the processes E_i. By definition the empty sum is δ.

We will give a model for process terms. Let ν be an instantiation of the variables in V with arbitrary processes. We give an interpretation M_ν of terms in $\Delta_{A,V}$ as follows. The term δ is represented as the process δ, so $M_\nu(\delta) = \delta$. For $x \in V$, $M_\nu(x) = \nu(x)$. For $a \in A_\tau$, $M_\nu(ap) = aM_\nu(p)$ and finally $M_\nu(p + q) = M_\nu(p) + M_\nu(q)$.

We give a special instantiation N of the variables in V with processes by setting $N(x) = [\{(p, x, p)\}]_p$, the process that can only do an x-labeled step to itself. So M_N is an interpretation of terms in $\Delta_{A,V}$ by processes with action set $A \cup V$. The following proposition shows that the interpretation M_N covers all other interpretations.

Proposition 4 $M_N(p) \preceq M_N(q) \Leftrightarrow \forall \nu : M_\nu(p) \preceq M_\nu(q)$

Proof: The "only if" part is trivial. Now choose a ν and let p and q possess the variables $x_1 \cdots x_n$. Then $M_\nu(p) = \partial_H(M_N(p) || p_1 || \cdots || p_n)$, where p_i is derived from $\nu(x_i)$ by renaming its actions a to a_i (where the a_i are brand new actions), the communication relation γ maps each pair (x_i, a_i) to a, and H contains all variables and new actions a_i. Similarly, $M_\nu(q) = \partial_H(M_N(q) || p_1 || \ldots || p_n)$. Since the preorder $\preceq$ is a precongruence for the operators used, we may conclude that $p \leq q$ implies $M_\nu(p) \preceq M_\nu(q)$. □

We will identify a term p with its standard representation $M_N(p)$ as a process and speak of the traces of a term, it being deterministic and so on. Note that we have thus obtained a model for open terms. The existing literature only treats closed terms in this way. The following theorem states that the axioms and deduction rules are sound.

Theorem 5 *Let u, v be $\Delta_{A,V}$ terms. Then $u \leq v \Rightarrow u \preceq v$.*

Proof: For the A axioms, a strong bisimulation can be constructed. For CS and IF, a weak bisimulation can be constructed and the root condition of $\preceq$ can be verified. For C, a relation between processes can be given as indicated in Figure 3. We move to soundness of the deduction rules. The rules TR and RE follow from the fact that $\preceq$ is a preorder. SU follows from Theorem 3 and IN from Proposition 4. □

As usual, completeness is more intricate. We define a "bar" operator that converts a term into a deterministic term with the same traces. The formal definition uses the auxiliary operator $I(\_)$ that gives the set of initial actions of a term. We will use this operator to derive some useful identities, eventually leading to a completeness proof.

Definition 6 *The determinism operator and $I(\_)$ are defined by the following equations that use parameters $x \in V$, $a \in A$ and term parameters u, v, w.*

$$\begin{array}{ll}
I(\delta) = \emptyset & \overline{\delta} = \delta \\
I(x) = \emptyset & \overline{x + u} = x + \overline{u} \\
I(au) = \{a\} & \overline{au + av + w} = \overline{a(u+v) + w} \\
I(\tau u) = I(u) & \overline{\tau u + v} = \overline{u + v} \\
I(u+v) = I(u) \cup I(v) & a \notin I(v) \Rightarrow \overline{au + v} = a\overline{u} + \overline{v}
\end{array}$$

We can eliminate the determinism operator from terms containing it by applying them from left to right as rewrite rules. Note that the instantiation rule IN can no longer be applied to variables $x \in V$, since they have become processes. Instead, we use the term parameters u, v, w. These can be instantiated. We now present a few identities with this operator.

Lemma 3 *For all terms t, u, v the following inequalities and equations can be derived.*

$$\begin{array}{llll}
a & \overline{u} \leq u & b & u = u + \overline{u} \\
c & \tau u = \tau\overline{u} + u & d & \tau\overline{u} + \tau\overline{(u+v)} = \tau\overline{u} + \overline{(u+v)} \\
e & \overline{(u+v)} + \overline{(v+w)} \leq \overline{u} + \overline{(v+w)} & f & \tau\overline{(u+v)} + \overline{(v+w)} \leq \tau\overline{u} + \overline{(v+w)}
\end{array}$$

Proof: Parts a, b and e are proved by structure induction as follows. A term u can be brought in one of the following forms: δ, $x + u'$, where $x \in V$, $\tau u' + u''$, or $au' + u''$, where $a \notin I(u'')$. The condition $a \notin I(w)$ can be achieved by applying axiom C if necessary. We use case analysis for the four cases. The induction hypothesis (IH) is that the statement holds for all terms that are simpler than the one that is being examined. So if we want to prove $P(u)$ in e.g. the case $u = au' + u''$, we may assume $P(u' + u'')$, $P(u')$ and $P(u'')$.

We prove inequality a : $\overline{u} \leq u$. The case $u = \delta$ is immediate. If $u = x + u'$, then $\overline{u} = \overline{x + u'} = x + \overline{u}' \leq^{IH} x + u' = u$. If $u = \tau u' + w$, then $\overline{\tau u' + u''} = \overline{u' + u''} \leq^{IH} u' + u'' \leq^{IF} \tau u' + u''$. Finally, $\overline{au' + u''} = a\overline{u}' + \overline{u}'' \leq^{IH} au' + u''$. A consequence: $\overline{(u+v)} = (\overline{u} + \overline{(u+v)} \leq \overline{u} + \overline{(u+v)}$.

Next, we prove $b : u = u + \overline{u}$. From a, we have $u + \overline{u} \leq u + u = u$. So we are left with proving $u \leq u + \overline{u}$. We proceed as above. The cases δ and $x + v$ are immediate. If $u = \tau v + w$, then $\tau v + w + \overline{\tau v + w} = \tau v + w + \overline{v + w} =^{T2} \tau v + v + w + \overline{v + w} =^{IH} \tau v + v + w =^{T2} \tau v + w$. Finally, $av + w + \overline{av + w} = av + w + a\overline{v} + w \geq^{C,IF} a(v + \overline{v}) + w =^{IH} av + w$.

We next prove $c : \tau u = \tau\overline{u} + u$. We have $\tau u =^{T2} \tau u + u \geq^{a} \tau\overline{u} + u$ and $\tau\overline{u} + u =^{CS} \tau(\tau\overline{u} + u) \geq^{IF} \tau(\overline{u} + u) =^{b} \tau u$.

We next prove $d : \tau\overline{u} + \tau\overline{(u + v)} = \tau\overline{u} + \overline{(u + v)}$. One side is immediate from IF; we prove the other. We have $\tau\overline{u} + \overline{(u + v)} =^{CS,A3} \tau\overline{u} + \tau(\tau\overline{u} + \overline{(u + v)}) \geq^{IF} \tau\overline{u} + \tau(\overline{u} + \overline{(u + v)}) \geq^{a} \tau\overline{u} + \overline{(u + v)}$.

We next prove $e \Rightarrow f$: If $\overline{(u + v)} + \overline{(v + w)} \leq u + \overline{(v + w)}$, then $\tau\overline{(u + v)} + \overline{(v + w)} \leq \tau u + \overline{(v + w)}$. We have $\tau\overline{(u + v)} + \overline{(v + w)} =^{T2} \tau\overline{(u + v)} + \overline{(u + v)} + \overline{(v + w)} \leq^{e} \tau\overline{(u + v)} + u + \overline{(v + w)} \leq^{IF} \tau\overline{(u + v)} + \tau u + \overline{(v + w)} =^{d} \overline{(u + v)} + \tau u + \overline{(v + w)} \leq^{e} u + \tau u + \overline{(v + w)} =^{T2} \tau u + \overline{(v + w)}$. By applying $T2$, from $\overline{(u + v)} + \overline{(v + w)} \leq u + \overline{(v + w)}$ we can even derive $\tau\overline{(u + v)} + \tau\overline{(v + w)} \leq \tau u + \tau\overline{(v + w)}$ and by adding C, $a\overline{(u + v)} + a\overline{(v + w)} \leq au + a\overline{(v + w)}$.

We now prove e by induction on the structure of v. The interesting case is $v = av' + v''$, where $a \notin I(v'')$. We consider subcases depending on the value of $a \in I(u)$ and $a \in I(w)$. The interesting subcase is when both conditions hold. So we may write $u = au' + u''$, $w = aw' + w''$, where $a \notin I(u'') \cup I(w'')$. So we must prove
$\overline{(au' + u'' + av' + v'')} + \overline{(av' + v'' + aw' + w'')} \leq au' + u'' + \overline{(av' + v'' + aw' + w'')}$.
This is rewritten to $a\overline{(u' + v')} + \overline{u'' + w''} + a\overline{(v' + w')} + \overline{v'' + w''} \leq au' + u'' + a\overline{(v' + w')} + \overline{v'' + w''}$. From IH, $\overline{u'' + w''} + \overline{v'' + w''} \leq u'' + \overline{v'' + w''}$ and, again from IH, using the $e \Rightarrow f$ derivation above, we have $a\overline{(u' + v')} + a\overline{(v' + w')} \leq au' + a\overline{(v' + w')}$. □

Note that u is stable and deterministic iff $u = \overline{u}$ and that $u \ll_{\mathrm{I}} v$ iff $\overline{v} = \overline{u + v}$.

Let $\sim = \preceq \cup \preceq^{-1}$. We define normal forms for terms modulo $\sim$. The general idea behind this normal form seems to be that choices are delayed and τ's skipped maximally without leaving the equivalence class. So $ax + ay$ is normalized to $a(\tau x + \tau y)$, delaying the choice until a has occurred and $x + \tau(y + \tau z)$ becomes $x + y + \tau z$, skipping the first τ.

There are three subclasses: *stable*, *pure* and *mixed* normal forms (pure and mixed being unstable). The stable normal form is - apart from variables - a sum of subterms au' in which all initial actions a differ. The pure normal form is τu, where u is stable. It is allowed only in the root; outside the root the initial τ of a pure term is skipped. The mixed form has a stable part u' and unstable parts τu_i where the u_i are deterministic. Some additional conditions are added to ensure that it is not equivalent to a pure term: the u_i must be mutually $\ll_{\mathrm{I}}$-incomparable and $\ll_{\mathrm{I}}$-majorated by u'. We shall show that each $\Delta_{A,V}$ term can be normalized, rewriting it modulo the axioms to a term in normal form.

Definition 7 *A $\Delta(A, V)$ term u is in stable normal form iff it can be represented as $\sum_{b \in B} bu'_b + \sum_{x \in X} x$, where $B \subseteq A$, $u'_b \in \mathcal{N}_M \cup \mathcal{N}_S$ for each $b \in B$ and $X \subseteq V$.*
It is in pure normal form iff it is $\tau u'$, where u' is in stable normal form.
It is in mixed normal form iff it can be represented as $\sum_{i \in I} \tau u_i + u'$, where u' is in stable normal

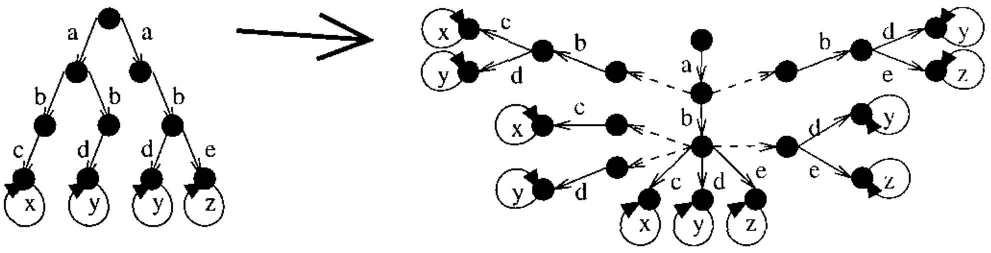
Figure 5: Normal form for $a(bcx + bdy) + ab(dy + ez)$

form and $\forall i \in I :: u_i = \overline{u_i} \neq \overline{u'} = \overline{u' + u_i}$. *and* $\forall i, j \in I : u_i = \overline{u_i + u_j} : i = j$. *It is in normal form iff it is in either stable, pure or mixed normal form.*

Lemma 4 *Let u be a $\Delta_{A,V}$ term. Then there exists a term v such that $u = v$ is derivable and v is in normal form.*

Proof: We give a normalization recipe. Write u as $\sum_I \tau u_i + u'$ where u' is stable. Replace every summand τu_i by the equivalent (Lemma 3*c*) summand $\tau\overline{u_i} + u_i$ Whenever $i, j \in I$ such that $i \neq j$ and $\overline{u_j} = \overline{u_j + u_i}$, replace $\tau\overline{u_i} + \tau\overline{u_j}$ by $\tau\overline{u_i} + \overline{u_j}$ (Lemma 3*d*). We obtain the term $\sum_J \tau v_j + v'$, where v' is stable and the v_j are deterministic and mutually $\ll_1$-incomparable. Also $v' = v' + v_j$ for any $j \in J$. If $v_j = \overline{v'}$ for some j, then $v_j = \overline{v'}$ for all j and replace (Lemma 3*c*) $\sum \tau v_j + v'$ by $\tau v'$, obtaining a pure term. Write v' as $\sum_{k \in K} a_k w_k + \sum_X x$ and whenever $a_k = a_l = b$ for some $k \neq l$ in K, replace (C) $bw_k + bw_l$ by $b(\tau w_k + \tau w_l)$ until all summands of v' have different initial actions. Repeat this process for the lower-order nodes of v; if such a lower-order node becomes pure, use equation T1 to skip the τ. $\square$

In Figure 5 the result of normalizing an example term is depicted. The τ-labeled edges are dashed. We now derive necessary conditions for $u \preceq v$ if u, v are normalized.

Lemma 5 *Let u, v be normalized and $u \preceq v$.*
If u stable then either v stable or $v = \tau v_0$ pure and $u \preceq v_0$.
If $u = \tau u_0$ pure then $v = \tau v_0$ pure and $u_0 \preceq v_0$.
If $u = \sum_I \tau u_i + u_0$ mixed then $v = \sum_J \tau v_j + v_0$ mixed and $u_0 \preceq v_0$ and for each $i \in I$ there exists a $j \in J$ such that $u_i = \overline{u_i + v_j}$.

Proof: The first two cases are immediate, so we assume the third. So $u = \sum_I \tau u_i + u_0$ and $v = \sum_J \tau v_j + v_0$ with u_0, v_0 stable. Suppose $u_0 \overset{\sigma}{\Longrightarrow} u'$. If σ is the empty trace, then $u' = u_0$ and by $\overline{u}_0 = \overline{u} = \overline{v} = \overline{v}_0$ we have $v_0 \overset{\sigma}{\Longrightarrow} v_0$ and $v_0 \ll_1 u_0$. For nonempty σ, we have $u \overset{\sigma}{\Longrightarrow} u'$, so since $u \preceq v$, there exists a v' such that $v \overset{\sigma}{\Longrightarrow} v'$ and $v' \ll_1 u'$, and since σ is nonempty, we have $\sum_J v_j + v_0 \overset{\sigma}{\Longrightarrow} v'$ and since v is normalized, $v_0 = \sum_J v_j + v_0$, so $v_0 \overset{\sigma}{\Longrightarrow} v'$. We have thus proven that $u_0 \ll_2 v_0$, and since u_0, v_0 are stable, $u_0 \preceq v_0$. We proceed with the second condition. Suppose $i \in I$. We then have $u \overset{\epsilon}{\Longrightarrow} u_i$. So there is a v' such that $v \overset{\epsilon}{\Longrightarrow} v'$ and $v' \ll_1 u_i$. Since u is unstable, by the root condition v is unstable too. If $v' = v$, then there exists a $j \in J$

and $v_j \ll_1 v \ll_1 u_i$. If $v' \neq v$, then, since the v_j are stable, $v' = v_j$ for some $j \in J$. □

Theorem 6 *Let u, v be $\Delta_{A,V}$ terms such that $u \preceq v$. Then $u \leq v$.*

Proof: We prove the theorem for normalized u, v first. We use structure induction. Suppose u is stable. If $v = \tau v'$ with v' stable and $u \preceq v'$, we use IH to derive $u \leq v'$ and by the IF rule $u \leq \tau v' = v$. If v is stable, we write $u = \sum_B bu_b + \sum_X x$ and $v = \sum_C cv_c + \sum_Y y$. Since u, v are trace equivalent, we deduce that $B = C$ and $X = Y$. Suppose $u_a \xRightarrow{\sigma} u'$ for some $a \in A$. Then $u \xRightarrow{a\sigma} u'$, so there exists a v' such that $v \xRightarrow{a\sigma} v'$ and $v' \ll_1 u'$. Since av_a is the only summand with initial action a, we find that $v_a \xRightarrow{\sigma} v'$. So $u_a \ll_2 v_a$. The $\ll_1$ and root condition present no problems, so $u_a \preceq v_a$. By IH, we derive $u_a \leq v_a$. By substitution this yields $u \leq v$. If u is pure the induction is immediate. So suppose $u = \sum_{i \in I} \tau u_i + u'$ and $v = \sum_{j \in J} \tau v_j + v'$ are mixed with u', v' stable. By Lemma 5, we have that $u' \preceq v'$, so by IH $u' \leq v'$. Let $i \in I$. By the same lemma there exists a $j \in J$ such that $u_i = \overline{u_i + v_j}$. By Lemma 3$f$, $\tau u_i + \overline{u'} = \tau \overline{u_i + v_j} + \overline{v_j + u'} \leq \tau v_j + \overline{v_j + u'} = \tau v_j + \overline{v'}$. By Lemma 3$b$, $u' = u' + \overline{u'}$ and $v' = v' + \overline{v'}$, so $\tau u_i + u' \leq \tau v_j + v'$. Now $v' = \sum_J v_j + v'$, so $\tau v_j + v' = \tau v_j + \sum_J v_j + v' \leq^I F \sum_J \tau v_j + v' = v$. We have proved $u' \leq v'$ and for each $i \in I$ $\tau u_i + u' \leq v$ so adding the summands and applying A3 yields $u \leq v$.

We drop the the restriction that u, v are normalized, so let $u \preceq v$ for general u, v. We normalize u, v to $\tilde{u}, \tilde{v}$ respectively. Since $u = \tilde{u}$, $v = \tilde{v}$, we have by soundness $u \sim \tilde{u}$, $v \sim \tilde{v}$, so transitivity of $\preceq$ yields $\tilde{u} \preceq \tilde{v}$. If this implies $\tilde{u} \leq \tilde{v}$, then transitivity of $\leq$ yields $u \leq v$. □

Due to the soundness theorem, the conditions in Lemma 5 are sufficient as well. So the algorithm in Lemma 4, together with the conditions in Lemma 5 as well as the technique sketched for the elimination of the determinism operator give a decision algorithm of $\preceq_2$ for $\Delta_(A, V)$ terms. However, this algorithm has a rather bad complexity, so we skip a further elaboration.

We now present an axiom and three derivation rules for the recursion operator. With the standard axioms for the other operators (cf. [4]), we can calculationally derive impossible future inclusion (albeit for open terms only). The KFAR (Koomen's Fair Abstraction) rule states that we can remove τ-loops from a process. There exist several generalizations of it. A corollary is obtained by taking $N = \emptyset$, giving $\tau^*\delta = \tau\delta$: livelock is deadlock.

Theorem 7 *Let I be an index set and let $M \subseteq (I \times A \times I)$, such that for any $i \in I$, the set $\{(a, j) \mid (i, a, j) \in M\}$ is finite. Then the rules in Table 4 are valid.*

Proof: The rule REC is a direct consequence of the defining SOS rule in Table 2. The RIP rules (Recursive Inequality Principles) can be derived in the "standard" way (cf. [3]) from AIP (Theorem 4). For KFAR it suffices to construct the obvious weak bisimulation between the processes concerned. □

$$[M]_k = \sum_{(k,a,j)\in M} a[M]_j \quad \text{REC} \qquad \frac{M = N \cup \{(i,a,i)\}, a \in H}{\tau_H[M]_i = \tau\tau_H[N]_i} \quad \text{KFAR}$$

$$\frac{\forall i \in I : x_i \leq \sum_{(i,a,j)\in M} ax_j}{x_k \leq [M]_k} \quad \text{RIPL} \qquad \frac{\forall i \in I : x_i \geq \sum_{(i,a,j)\in M} ax_j}{x_k \geq [M]_k} \quad \text{RIPR}$$

Table 4: Recursion rules

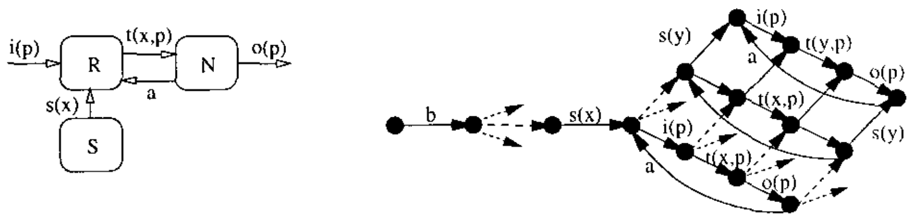

Figure 6: Mobile phone example: architecture and process

6 Example

We model and analyze some mobile telephony protocols. See Figure 6 for an illustration. The mobile phone network N consists of a large number of nodes. A mobile phone possesses a selector S that continuously determines the node to be connected to and a router R that communicates with the network nodes.

We have sets P of possible packets and X of possible nodes. For $p \in P$ and $x \in X$ oullset of actions S consists of

b	power on,
$i(p)$	accept input packet p,
$t(x, p)$	transmit packet p to node x,
$o(p)$	offer output packet p,
a	acknowledge transmission,
$s(x)$	select node x.

Our alphabet A consists of $\bigcup_{s\in S}\{s, s!, s?\}$: actions possibly decorated with a question or exclamation mark. Our communication function γ consists of $\bigcup_{s\in S}\{(s?, s!, s), (s!, s?, s)\}$. The connotation is that $s!$ represents sending, $s?$ receiving and s their synchronization. We encapsulate or block decorated actions: $H = \bigcup_{s\in S}\{s?, s!\}$ and hide the communications $I = \bigcup_{p\in P, x\in X}\{t(x, p), s(x), a\}$.

Then our telephone network T is given by $T = \tau_I(\partial_H(R||N||S))$, where R, N, S satisfy the following recursive equations.

$$\begin{array}{lcl}
S & = & bS' \\
S' & = & \sum_{x\in X} \tau s(x)!S' \\
R & = & \sum_{x\in X} s(x)?R_x \\
R_x & = & \sum_{p\in P} i(p)R_x^p + \sum_{y\in X} s(y)?R_y \\
R_x^p & = & t(x,p)!R_x' + \sum_{y\in X} s(y)?R_y^p \\
R_x' & = & a?R_x + \sum_{y\in X} s(y)?R_y' \\
N & = & \sum_{p\in P, x\in X} t(x,p)?(o(p)a\delta||N)
\end{array}$$

In Figure 6 the essential states and transitions of $\partial_H(R||N||S)$ are shown. We deduce that T satisfies the following recursive equations.

$T = bT', T' = \sum_{p\in P} i(p)o(p)T'$.

Clearly, T is deterministic, so this deduction holds in every preorder between $\preceq_2$ and branching bisimilarity. However, note that the axiom CS allows us to obtain these simple equations for S. If we use branching or even weak bisimilarity, the order in which the signals from nearby nodes are treated by the selector does influence its protocol. Note that the terms $\tau x + \tau(\tau y + \tau z)$ and $\tau x + \tau y + \tau z$ are only the same in coupled simulation and weaker preorders. Nevertheless, since the final result is deterministic, the implementation of S does not affect the outcome even in branching bisimilarity.

We may assume a new model composed of $\mathcal{R}$ and $\mathcal{S}$. The new selector $\mathcal{S}$ can select more than one node. This added feature enables $\mathcal{R}$ to select a preferred node if possible. Thus $\mathcal{S}$ satisfies $\mathcal{S} = b\mathcal{S}'$, $\mathcal{S}' = \sum_{\mathcal{X}\subseteq X} \tau \sum x \in \mathcal{X} s(x)\mathcal{S}'$. Now we have $\mathcal{S} \preceq_2 S$, from $\tau(x+y) \preceq^I F\tau(\tau x + \tau y) =^C S\tau x + \tau y$. Therefore, $\tau_I(\partial_H(R||N||\mathcal{S})) \preceq_2 T$, and since T is deterministic, they are even branching bisimilar.

We conclude that the new selector can replace the old one in the old model without compromising its functionality. If the new selector has about the same price as the old one, this observation can save a lot of storage and production costs.

7 Conclusion and further work

Many preorders (including equivalence relations) used for specification and verification are based upon some notion of observability. However, many liveness properties that are vital for the specification of certain systems are not observable. On the other hand, bisimulation based preorders often make unnecessary distinctions between processes, thus restricting implementer freedom.

In this paper we define classes of safety and liveness notions that disregard the branching behavior of processes to some extent and process preorders $\preceq_n$ that go with them. The preorder $\preceq_2^r$ has an attractive-looking axiomatization. We have shown how this preorder is related to the concept of determinism. It can be used to specify the "maximally allowed nondeterminism" of a system.

There is, however, a price to pay for implementer freedom gained from using $\preceq_2^r$: verifications become much harder computationally than with bisimilarity. Derivations use both the special nature of the silent step and the asymmetry of the preorder.

The preorder $\preceq_2^r$ is the weakest we know of that is a congruence w.r.t. the CSP/ACP operators, that satisfy the approximation induction principle (AIP) and that distinguishes good protocols from bad ones like in Figure 1. An interesting open problem is to determine whether there exist weaker preorders that satisfy these conditions. It also seems interesting to investigate "stability" in the same way as we did with determinism.

References

[1] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Proceedings ATPN '97*, volume 1248 of *Lecture Notes in Computer Science*, Toulouse, France, 1997. Springer–Verlag, Berlin.

[2] J.C.M. Baeten and S. Mauw. Delayed choice: an operator for joining Message Sequence Charts. In D. Hogrefe and S. Leue, editors, *Formal Description Techniques VII*, pages 340–354. Kluwer Academic Publishers, Boston, 1995.

[3] J.C.M. Baeten and C. Verhoef. Concrete Process Algebra. In A. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4, pages 149–268. Oxford University Press, Clarendon, UK, 1995.

[4] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1990.

[5] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation Can't Be Traced. *Journal of the ACM*, 42(1):232–268, 1995.

[6] R. De Nicola and M. Hennessy. Testing Equivalences for Processes. *Theoretical Computer Science*, 34:83–133, 1984.

[7] J. Engelfriet. Determinacy implies observation equivalence = trace equivalence. *Theoretical Computer Science*, 36:21–25, 1985.

[8] R.J. van Glabbeek. The Linear Time - Branching Time Spectrum II. In E. Best, editor, *Proceedings CONCUR '93*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer–Verlag, Berlin, 1993. Full version: *ftp://Boole.stanford.edu/pub/spectrum.ps.gz*.

[9] R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics (extended abstract). In G.X. Ritter, editor, *Proceedings IFIP '89*, pages 613–618. North Holland, Amsterdam, 1989.

[10] M. Hennesy and R. Milner. Algebraic Laws for Nondeterminism and Concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[11] J. Parrow and P. Sjödin. Multiway Synchronization Verified with Coupled Simulation. In W.R. Cleaveland, editor, *Proceedings CONCUR '92*, volume 630 of *Lecture Notes in Computer Science*, pages 518–533. Springer–Verlag, Berlin, 1992.

[12] W.C. Rounds and S.D. Brookes. Possible Futures, Acceptances, Refusals and Communicating Processes. In *Proceedings 22-nd Annual Symposium on Foundations of Computer Science*, pages 140–149. IEEE, 1981.

97/08	P. Hoogendijk and R.C. Backhouse	When do datatypes commute? p. 35.
97/09	Proceedings of the Second International Workshop on Communication Modeling, Veldhoven, The Netherlands, 9-10 June, 1997.	Communication Modeling- The Language/Action Perspective, p. 147.
97/10	P.C.N. v. Gorp, E.J. Luit, D.K. Hammer E.H.L. Aarts	Distributed real-time systems: a survey of applications and a general design model, p. 31.
97/11	A. Engels, S. Mauw and M.A. Reniers	A Hierarchy of Communication Models for Message Sequence Charts, p. 30.
97/12	D. Hauschildt, E. Verbeek and W. van der Aalst	WOFLAN: A Petri-net-based Workflow Analyzer, p. 30.
97/13	W.M.P. van der Aalst	Exploring the Process Dimension of Workflow Management, p. 56.
97/14	J.F. Groote, F. Monin and J. Springintveld	A computer checked algebraic verification of a distributed summation algorithm, p. 28
97/15	M. Franssen	λP-: A Pure Type System for First Order Loginc with Automated Theorem Proving, p.35.
97/16	W.M.P. van der Aalst	On the verification of Inter-organizational workflows, p. 23
97/17	M. Vaccari and R.C. Backhouse	Calculating a Round-Robin Scheduler, p. 23.
97/18	Werkgemeenschap Informatiewetenschap redactie: P.M.E. De Bra	Informatiewetenschap 1997 Wetenschappelijke bijdragen aan de Vijfde Interdisciplinaire Conferentie Informatiewetenschap, p. 60.
98/01	W. Van der Aalst	Formalization and Verification of Event-driven Process Chains, p. 26.
98/02	M. Voorhoeve	State / Event Net Equivalence, p. 25
98/03	J.C.M. Baeten and J.A. Bergstra	Deadlock Behaviour in Split and ST Bisimulation Semantics, p. 15.
98/04	R.C. Backhouse	Pair Algebras and Galois Connections, p. 14
98/05	D. Dams	Flat Fragments of CTL and CTL*: Separating the Expressive and Distinguishing Powers. P. 22.
98/06	G. v.d. Bergen, A. Kaldewaij V.J. Dielissen	Maintenance of the Union of Intervals on a Line Revisited, p. 10.
98/07	Proceedings of the workhop on Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98) June 22, 1998 Lisbon, Portugal	edited by W. v.d. Aalst, p. 209
98/08	Informal proceedings of the Workshop on User Interfaces for Theorem Provers. Eindhoven University of Technology ,13-15 July 1998	edited by R.C. Backhouse, p. 180
98/09	K.M. van Hee and H.A. Reijers	An analytical method for assessing business processes, p. 29.
98/10	T. Basten and J. Hooman	Process Algebra in PVS
98/11	J. Zwanenburg	The Proof-assistemt Yarrow, p. 15
98/12	Ninth ACM Conference on Hypertext and Hypermedia Hypertext '98 Pittsburgh, USA, June 20-24, 1998 Proceedings of the second workshop on Adaptive Hypertext and Hypermedia.	Edited by P. Brusilovsky and P. De Bra, p. 95.
98/13	J.F. Groote, F. Monin and J. v.d. Pol	Checking verifications of protocols and distributed systems by computer. Extended version of a tutorial at CONCUR'98, p. 27.
98/14	T. Verhoeff (artikel volgt)	
99/01	V. Bos and J.J.T. Kleijn	Structured Operational Semantics of χ , p. 27
99/02	H.M.W. Verbeek, T. Basten and W.M.P. van der Aalst	Diagnosing Workflow Processes using Woflan, p. 44

99/03	R.C. Backhouse and P. Hoogendijk	Final Dialgebras: From Categories to Allegories, p. 26
99/04	S. Andova	Process Algebra with Interleaving Probabilistic Parallel Composition, p. 81
99/05	M. Franssen, R.C. Veltkamp and W. Wesselink	Efficient Evaluation of Triangular B-splines, p. 13
99/06	T. Basten and W. v.d. Aalst	Inheritance of Workflows: An Approach to tackling problems related to change, p. 66
99/07	P. Brusilovsky and P. De Bra	Second Workshop on Adaptive Systems and User Modeling on the World Wide Web, p. 119.
99/08	D. Bosnacki, S. Mauw, and T. Willemse	Proceedings of the first international syposium on Visual Formal Methods - VFM'99
99/09	J. v.d. Pol, J. Hooman and E. de Jong	Requirements Specification and Analysis of Command and Control Systems
99/10	T.A.C. Willemse	The Analysis of a Conveyor Belt System, a case study in Hybrid Systems and timed μ CRL, p. 44.
99/11	J.C.M. Baeten and C.A. Middelburg	Process Algebra with Timing: Real Time and Discrete Time, p. 50.
99/12	S. Andova	Process Algebra with Probabilistic Choice, p. 38.
99/13	K.M. van Hee, R.A. van der Toorn, J. van der Woude and P.A.C. Verkoulen	A Framework for Component Based Software Architectures, p. 19
99/14	A. Engels and S. Mauw	Why men (and octopuses) cannot juggle a four ball cascade, p. 10
99/15	J.F. Groote, W.H. Hesselink, S. Mauw, R. Vermeulen	An algorithm for the asynchronous *Write-All* problem based on process collision*, p. 11.
99/16	G.J. Houben, P. Lemmens	A Software Architecture for Generating Hypermedia Applications for Ad-Hoc Database Output, p. 13.
99/17	T. Basten, W.M.P. v.d. Aalst	Inheritance of Behavior, p.83
99/18	J.C.M. Baeten and T. Basten	Partial-Order Process Algebra (and its Relation to Petri Nets), p. 79
99/19	J.C.M. Baeten and C.A. Middelburg	Real Time Process Algebra with Time-dependent Conditions, p.33.
99/20	Proceedings Conferentie Informatiewetenschap 1999 Centrum voor Wiskunde en Informatica 12 november 1999, p.98	edited by P. de Bra and L. Hardman
00/01	J.C.M. Baeten and J.A. Bergstra	Mode Transfer in process Algebra, p. 14
00/02	J.C.M. Baeten	Process Algebra with Explicit Termination, p. 17.
00/03	S. Mauw and M.A. Reniers	A process algebra for interworkings, p. 63.
00/04	R. Bloo, J. Hooman and E. de Jong	Semantical Aspects of an Architecture for Distributed Embedded Systems*, p. 47.
00/05	J.F. Groote and M.A. Reniers	Algebraic Process Verification, p. 65.
00/06	J.F. Groote and J. v. Wamel	The Parallel Composition of Uniform Processes wit Data, p. 19
00/07	C.A. Middelburg	Variable Binding Operators in Transition System Specifications, p. 27.
00/08	I.D. van den Ende	Grammars Compared: A study on determining a suitable grammar for parsing and generating natural language sentences in order to facilitate the translation of natural language and MSC use cases, p. 33.
00/09	R.R. Hoogerwoord	A Formal Development of Distributed Summation, p. 35
00/10	T. Willemse, J. Tretmans and A. Klomp	A Case Study in Formal Methods: Specification and Validation on the OM/RR Protocol, p. 14.
00/11	T. Basten and D. Bošnački	Enhancing Partial-Order Reduction via Process Clustering, p. 14
00/12	S. Mauw, M.A. Reniers and T.A.C. Willemse	Message Sequence Charts in the Software Engineering Process, p. 26
00/13	J.C.M. Baeten, M.A. Reniers	Termination in Timed Process Algebra, p. 36