The Use of Static Constructs in A Modal Process Logic

Hans Hüttel
Laboratory for the Foundations of Computer Science, JCMB
University of Edinburgh, Edinburgh EH9 3JZ
SCOTLAND

Kim G. Larsen
Department of Mathematics and Computer Science
Aalborg University Centre, Strandvejen 19, DK-9000 Aalborg
DENMARK

1 Introduction

In [8] Kim Larsen and Bent Thomsen introduced a modal process logic, MPL, intended for specification of nondeterministic and concurrent processes expressed in some Process Algebra, e.g. $\mathcal{CCS}$. Over the years, several equivalences between non–deterministic and concurrent processes have been proposed in order to capture various observational aspects of processes [9, 10]. This has allowed Process Algebra itself to be used as a specification language with the notion of correctness of an implementation with respect to a specification being simply that of their equivalence. However, practical experience [6, 7] has demonstrated the need for *loose specifications*: i.e. specifications which may be satisfied by more than a single equivalence class of processes. This observation was the motivation for the introduction of MPL, a logic allowing such loose specifications to be expressed.

The logic MPL is an extension of Process Algebra in the sense that the connectives in the logic for building formulae include the process constructs from Process Algebra. Moreover, the formulae of the logic are given an *operational semantics* capturing the behaviour of the possible implementations. Based on this semantics a *refinement ordering* between formulae is defined, extending the well–established notion of bisimulation equivalence [11, 10]. The refinement ordering allows a comparison of specifications with respect to their looseness in terms of the implementations they allow, and since MPL subsumes Process Algebra it can also be used for comparing an implementation with a specification.

In [8, 3] the theoretical foundations of MPL were given. In particular results on full abstraction were given, establishing equivalences between operational and denotational semantics of MPL. Also, it was shown that — from a *theoretical* point of view — MPL is less expressive than Hennessy–Milner Logic (with recursion added) [1, 5]. In this paper we want to demonstrate that — from a *practical* point of view — MPL suffices. Thus, a number of examples has been worked out, including a data link (a simple protocol), a restartable system, and a simple scheduler. In all the examples, the looseness of MPL plays a central — though varying — role.

If one wants to model realistic systems conveniently one needs a number of *static constructs*, in particular parallel composition of components. In this paper we make extensive use of the constructs of $\mathcal{CCS}$ (especially parallel composition) on MPL formulae. This allows us to describe a wide variety of systems, but it also introduces the possibility of "unobservable" transitions. It is therefore desirable to introduce an *observational refinement ordering* disregarding these transitions. To support compositionality of the verification process (i.e. the ability to deduce correctness properties of combined processes from correctness properties of the components) we investigate to what extent the observational refinement ordering is preserved by the constructs of MPL.

2 Modal transition systems

MPL is given an operational description in the style of [13], using labelled transition systems. Specifications impose restrictions on the transitions of possible implementations by telling which ones they *must* and *may* have. The transition systems for MPL specifications therefore have *two* transition relations: $\rightarrow_\Box$ describing the *required* transitions and $\rightarrow_\Diamond$ describing the *allowed* transitions:

Definition 1 *A* modal transition system $\mathcal{S} = (Sp, Act, \rightarrow_\Diamond, \rightarrow_\Box)$ *is a quadruple where Sp is the set of* specifications, *Act is the* action *set and* $\rightarrow_\Diamond$ *and* $\rightarrow_\Box$ *are* modal transition relations, *i.e.* $\rightarrow_\Box \subseteq Sp \times Act \times Sp$, $\rightarrow_\Diamond \subseteq Sp \times Act \times Sp$ *satisfying the condition* $\rightarrow_\Box \subseteq \rightarrow_\Diamond$

The last clause says that anything required is also allowed. We also assume a set of *implementations* or *processes*, with an operational semantics defined through a standard labelled transition system $\mathcal{P} = (Pr, Act, \rightarrow)$ with identical action set. It can be seen as a modal transition system with $\rightarrow = \rightarrow_\Diamond = \rightarrow_\Box$.

The concept of *refinement* extends that of bisimulation equivalence [10]. An MPL specification S is a refinement of another specification T when all transitions *allowed* by S are also allowed by T. Conversely, all transitions *required* by T must be required by S. Furthermore, in both cases the results of matching transitions are again in the relation. More formally we make the following definition:

Definition 2 *A relation* $R \subseteq Sp \times Sp$ *on a modal transition system* $\mathcal{S} = (Sp, Act, \rightarrow_\Diamond, \rightarrow_\Box)$ *is a* refinement *iff whenever* SRT *the following holds:*

1. *Whenever* $S \xrightarrow{a}_\Diamond S'$, *then for some* T', $T \xrightarrow{a}_\Diamond T'$ *with* $S'RT'$
2. *Whenever* $T \xrightarrow{a}_\Box T'$, *then for some* S', $S \xrightarrow{a}_\Box S'$ *with* $S'RT'$

S is said to be a refinement of T, when (S, T) is contained in some refinement R, and we write $S \lhd T$.

As is the case for bisimulation equivalence [10], $\lhd$ is the union of all refinements, it is itself a refinement and a *preorder*. When $\rightarrow_\Diamond = \rightarrow_\Box$ (as e.g. in a process system) $\lhd$ collapses to bisimulation equivalence, $\sim$.

We can compare specifications from different modal transition systems by taking their disjoint union. When $p \lhd T$, with p being a process and T being a specification, we say that p is an *implementation* of T. The weakest specification $\mathcal{U}$ — satisfied by any process — constantly allows any action, but never guarantees that any action must be performed. Operationally, $\mathcal{U}$ is completely defined by $\mathcal{U} \xrightarrow{a}_\Diamond \mathcal{U}$ for all $a \in Act$.

3 MPL with static constructs

The parallel composition and action restriction operators of $\mathcal{CCS}$ [9] are just special cases of the more general idea of static constructs, so called because they describe a fixed linkage between a set of components. Every static construct associates a combination of actions performed by its components with an action performed by the entire construct. Thus one may regard a static construct as defined by a function over the set of actions. We introduce a special no–action symbol not in Act, 0, since some components may not contribute to the common action (as is the case for parallel composition). Letting $Act_0 = Act \cup \{0\}$, any static construct involving n components, $F_1, \ldots, F_n$, can be written as

$$(F_1, \ldots, F_n)[f]$$

where f is an n–ary partial function on actions, $f : Act_0^n \hookrightarrow Act$. f is partial since certain combinations of actions may be disallowed.

The operational rule for static constructs expresses that the construct has a transition just in case its components have the transitions required by f:

Definition 3 (Operational semantics of MPL) *Let $\mathcal{S} = (Sp, Act, \rightarrow_\Diamond, \rightarrow_\Box)$ be the modal transition system where Sp is the language with the abstract syntax*

$$F ::= Nil \mid a_\Box.F_1 \mid a_\Diamond.F_1 \mid F_1 + F_2 \mid x \mid \mathrm{rec}x.F \mid (F_1, \ldots, F_n)[f]$$

with $f : Act_0^n \hookrightarrow Act$ and $\rightarrow_\Box$ and $\rightarrow_\Diamond$ are the least relations satisfying the operational rules and axioms in Table 1.

The introduction of certain static constructs may require assumptions about the structure of the action set. For the parallel composition of [9] we define Act to have the structure:

$$Act = A \cup \overline{A} \cup \{\tau\}$$

with $\overline{A}$ being the *co–names* of A. The co–name bijection $\overline{} : A \leftrightarrow \overline{A}$ has the usual property $\overline{\overline{a}} = a$ with the obvious extension to sets of actions.

As usual, think of $\overline{a}$–actions as representing outputs and 'unbarred' a–actions as actions representing inputs, the action τ being analogous to τ in $\mathcal{CCS}$, representing internal communications. Parallel composition may then be defined as:

$$S_1 \mid S_2 = (S_1, S_2)[f_{comp}]$$

with $f_{comp} : Act_0^2 \hookrightarrow Act$ defined by $f_{comp}(a, \overline{a}) = \tau$ and $f_{comp}(0, a) = f_{comp}(a, 0) = a$.

Similarly, the action restriction of [9] is :

$$S \setminus A = S[f_{res}^A]$$

where $f_{res}^A : Act_0 \hookrightarrow Act$ is defined by $f_{res}^A(a) = a$ if $a \in A$.

The restricted composition of [9] is defined by

$$S_1 \parallel S_2 = (S_1 \mid S_2) \setminus A$$

with A being the actions used for communication between S_1 and S_2.

Now, $\lhd$ *is* a precongruence with respect to *all* static constructs, ensuring compositionality:

$\Diamond$ -prefixing	$a_\Diamond.E \xrightarrow{a}_\Diamond E$
$\Box$ -prefixing	$a_\Box.E \xrightarrow{a}_\Box E$ $a_\Box.E \xrightarrow{a}_\Diamond E$
Nondeterminism	$\dfrac{E \xrightarrow{a}_m E'}{E + F \xrightarrow{a}_m E'}$ $\dfrac{F \xrightarrow{a}_m F'}{E + F \xrightarrow{a}_m F'}$
Recursion	$\dfrac{E[\mathrm{rec}x.E/x] \xrightarrow{a}_m E'}{\mathrm{rec}x.E \xrightarrow{a}_m E'}$
Static constructs	$\dfrac{\left[(F_i \xrightarrow{a_i}_m F_i') \ \vee \ (F_i = F_i' \wedge a_i = 0)\right]_{i \leq n}}{(F_1 \ldots F_n)[f] \xrightarrow{a}_m (F_1' \ldots F_n')[f]} \quad a \asymp f(a_1, \ldots, a_n)$

Table 1: Operational semantics of MPL. $m \in \{\Box, \Diamond\}$, and $a \asymp f(a_1, \ldots, a_n)$ is *true* if $f(a_1, \ldots, a_n)$ is defined and has the value a and is *false* otherwise.

Theorem 1 *If $F_i \lhd G_i$ for $i \leq n$, for any static construct $(\ldots)[f]$ we have*

$$(F_1, \ldots, F_n)[f] \lhd (G_1, \ldots, G_n)[f]$$

PROOF: We show that the relation

$$\{\langle (F_1 \ldots, F_n)[f], (G_1, \ldots, G_n)[f] \rangle \mid \forall i. F_i \lhd G_i\}$$

is a refinement. Suppose $(F_1, \ldots, F_n)[f] \xrightarrow{a}_\Diamond (F_1', \ldots, F_n')[f]$. (These are the only transitions possible according to the operational rule). By the operational rule for static constructs we have a set $S \subseteq \{1, \ldots, n\}$ such that $F_j \xrightarrow{a_j}_\Diamond F_j'$ when $j \in S$ and $F_j \xrightarrow{0}_\Diamond F_j$ otherwise. But then we have for the same S that $G_j \xrightarrow{a_j}_\Diamond G_j'$ whenever $j \in S$ with $F_j' \lhd G_j'$ and $G_j \xrightarrow{0}_\Diamond G_j$ otherwise. But this implies that we also have $\langle (F_1', \ldots, F_n')[f], (G_1', \ldots, G_n')[f] \rangle$ in the relation. The other half of the proof, concerning the transition $(G_1, \ldots, G_n)[f] \xrightarrow{a}_\Box (G_1', \ldots, G_n')[f]$, is similar. $\Box$

4 An observational refinement order

The simple data link P in Figure 1, first described in [12], lets information pulses pass through one at a time. Information enters on an a–action, the sender S transmits it to the medium M through a c–communication. The medium M may lose the message but then lets S know via a d–communication. M transmits the information to the receiver R which emits the information using

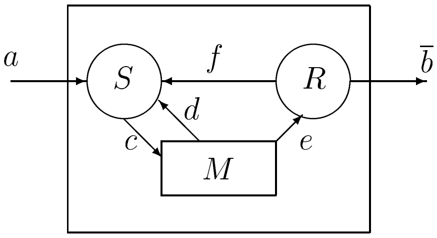

Figure 1: A simple data link P

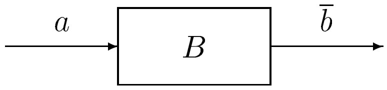

Figure 2: A 1–place buffer B

a $\overline{b}$–action, sending an acknowledgment to S using an f–communication. Note, that this system may *diverge* by information constantly getting lost and retransmitted.

The 1–place buffer B pictured in Figure 2 treats pulses one at a time; pulses enter on a and are emitted on $\overline{b}$.

It is thus an abstract description of the protocol, and we want a refinement ordering, $\trianglelefteq$, which reflects this by disregarding the unobservable behaviour of P, such that $P \trianglelefteq B$ holds.

$$P \trianglelefteq B$$

$\trianglelefteq$ is defined using two 'observational' transition relations on action *strings*, $\Longrightarrow_\diamond$ and $\Longrightarrow_\Box$. Let $\overset{\epsilon}{\Longrightarrow}_m$ $(m = \Box, \diamond)$ be the relexive and transitive closure of $\overset{\tau}{\rightarrow}_m$. Then for an observable string $w = a_1, \ldots, a_n$ (i.e. $a_i \neq \tau$ for all i), $S \overset{w}{\Longrightarrow}_m T$ if and only if $S \overset{\epsilon}{\Longrightarrow}_m \overset{a_1}{\rightarrow}_m \overset{\epsilon}{\Longrightarrow}_m \cdots \overset{\epsilon}{\Longrightarrow}_m \overset{a_n}{\rightarrow}_m \overset{\epsilon}{\Longrightarrow}_m T$, where juxtaposition denotes relation composition. Thus, $S \overset{w}{\Longrightarrow}_m T$ means that one by m–transitions corresponding to the action sequence w, possibly interleaved by τ_m–transitions, from S can derive T.

A relation $R \subseteq Sp \times Sp$ on $\mathcal{S} = (Sp, Act, \rightarrow_\diamond, \rightarrow_\Box)$ is an *observational refinement*, if R is a refinement in to the derived "observational" modal transition system $\mathcal{S}_\mathcal{W} = (Sp, (Act \setminus \{\tau\})^*, \Longrightarrow_\diamond, \Longrightarrow_\diamond)$. We write $S \trianglelefteq T$ whenever (S, T) is contained in some observational refinement R. It is easily seen that any ordinary refinement also is an observational refinement. Notice also that our definition of $\trianglelefteq$ subsumes the definition of observational equivalence $\approx$ [10]. Since the definition of observational refinement is stated in terms of string transitions, it is does not seem to be directly applicable from a practical point of view. However, it turns out that we can make do with action strings with length ≤ 1, as stated in the following proposition:

Proposition 1 *R is an observational refinement iff whenever SRT and $a \in Act$, then the following holds:*

1. $S \overset{a}{\rightarrow}_\diamond S' \Rightarrow \exists T' : T \overset{\phi(a)}{\Longrightarrow}_\diamond T' \wedge S'RT'$

2. $T \xrightarrow{a}_{\Box} T' \Rightarrow \exists S' : S \overset{\phi(a)}{\Longrightarrow}_{\Box} S' \wedge S'RT'$

where $\phi(\tau) = \epsilon$ *and* $\phi(a) = a$ *for* $a \neq \tau$.

$\trianglelefteq$ is *not* in general preserved by the static constructs. To see this, consider $f : Act_0 \hookrightarrow Act$ satisfying $f(\tau) = a \neq \tau$. Then, $\tau_{\Diamond}.Nil \trianglelefteq Nil$ but $(\tau_{\Diamond}.Nil)[f] \not\trianglelefteq (Nil)[f]$. To ensure preservation of $\trianglelefteq$, we must restrict the way in which the construct may utilize the τ–transitions of its components. A restricted class of static constructs preserving $\trianglelefteq$ is given by the following notion of strict τ–insensitivity.

Definition 4 *A function* $Act_0{}^n \hookrightarrow Act$ *is called* strictly τ-insensitive *if it satisfies the two conditions:*

1. $f(a_1, \ldots, 0, \ldots, a_n) \asymp f(a_1, \ldots, \tau, \ldots, a_n)$
2. $f(\tau, \ldots, \tau) = \tau$

The first condition in the above definition basically says that a τ-action by a component is disregarded by the construct. Unfortunately this definition of τ-insensitivity does not embrace parallel composition, since $f_{comp}(\tau, 0) \asymp \tau$ whereas $f_{comp}(\tau, \tau)$ is undefined. In a sense parallel composition is τ-sensitive since it does not allow internal communications among some components while other components are acting. To take this into account we loosen the definition a bit, getting:

Definition 5 *A function* $Act_0{}^n \hookrightarrow Act$ *is called* τ-insensitive *if it satisfies the two conditions:*

1. $f(a_1, \ldots, \tau, \ldots, a_n) \precsim f(a_1, \ldots, 0, \ldots, a_n)$ *if at least one* $a_i \neq 0$
2. $f(0, \ldots, \tau, \ldots, 0) = \tau$

where $t \precsim s$ *iff* s *is defined whenever* t *is and then has the same value.*

Notice that we here only require that *one* τ by a component leads to a τ performed by the construct. Certainly, f_{comp}, is τ-insensitive. It is also easy to see that any strictly τ-insensitive function is τ-insensitive.

Silent transitions by a component in a τ-insensitive construct lead to a silent transition by the entire construct, as can be seen by the following

Lemma 1 *If* $f : Act_0^n \hookrightarrow Act$ *is* τ*-insensitive, then whenever* $H \overset{\epsilon}{\Longrightarrow}_m H'$,

$$(E_1, \ldots, H, \ldots, E_n)[f] \overset{\epsilon}{\Longrightarrow}_m (E_1, \ldots, H', \ldots, E_n)[f]$$

PROOF: Trivial.

It turns out that $\trianglelefteq$ is a precongruence for τ-insensitive constructs.

Theorem 2 (Precongruence of $\trianglelefteq$) *Let* $f : Act_0^n \hookrightarrow Act$ *be* τ*-insensitive and* $F_i \trianglelefteq G_i$ *for* $1 \leq i \leq n$. *Then* $(F_1, \ldots, F_n)[f] \trianglelefteq (G_1, \ldots, G_2)[f]$.

PROOF: We show that

$$\mathbf{R} = \{\langle (F_1,\ldots,F_n)[f], (G_1,\ldots,G_n)[f]\rangle \mid F_i \trianglelefteq G_i, 1 \leq i \leq n\}$$

is an observational refinement (using Proposition 1).

So suppose

$$\langle (F_1,\ldots,F_n)[f], (G_1,\ldots,G_n)[f]\rangle \in \mathbf{R}$$

and that

$$(F_1,\ldots,F_n)[f] \xrightarrow{a}_\diamond (F_1',\ldots,F_n')[f]$$

Assume without loss of generality that the first j components $F_1,\ldots,F_j$ contribute with observable actions $a_1,\ldots.a_j$, that the next $k-j, k \geq 0$ components $F_{j+1},\ldots,F_k$ actions contribute with τ's and that the last components $F_{k+1},\ldots,F_n$ are inactive, i.e.

$$\begin{array}{ll} F_i \xrightarrow{a_i}_\diamond F_i' & {\scriptstyle 1\leq i\leq j} \\ F_i \xrightarrow{\tau}_\diamond F_i' & {\scriptstyle j+1\leq i\leq k} \\ F_i \xrightarrow{0}_\diamond F_i & {\scriptstyle k+1\leq i\leq n} \end{array}$$

with $f(a_1,\ldots,a_j,\tau,\ldots,\tau,0,\ldots,0) \asymp a$. Now, since $F_i \trianglelefteq G_i$ we can find matching transitions:

$$\begin{array}{ll} G_i \xRightarrow{\epsilon}_\diamond G_i'' \xrightarrow{a_i}_\diamond G_i''' \xRightarrow{\epsilon}_\diamond G_i' & {\scriptstyle 1\leq i\leq j} \\ G_i \xRightarrow{\epsilon}_\diamond G_i' & {\scriptstyle j+1\leq i\leq k} \\ G_i \xrightarrow{0}_\diamond G_i & {\scriptstyle k+1\leq i\leq n} \end{array}$$

Using Lemma 1 k times we obtain the transition

$$\begin{array}{l}(G_1,\ldots,G_j,G_{j+1},\ldots,G_k,G_{k+1},\ldots,G_n)[f] \xRightarrow{\epsilon}_\diamond \\ \quad (G_1'',\ldots,G_j'',G_{j+1}',\ldots,G_k',G_{k+1},\ldots,G_n)[f]\end{array}$$

If $j=0$, this is the matching transition since we by the τ-insensitivity of f have that $a \asymp f(a_1,\ldots,a_j,\tau,\ldots,\tau,0,\ldots,0) \asymp f(\tau,\ldots,\tau,0,\ldots,0) \asymp \tau$ since $k \geq 1$ (not all components are inactive). Otherwise $a \asymp f(a_1,\ldots,a_j,\tau,\ldots,\tau,0,\ldots,0) \precapprox f(a_1,\ldots,a_j,0,\ldots,0,\ldots,0)$. By the first condition in Definition 5 we get the transition

$$\begin{array}{l}(G_1'',\ldots,G_j'',G_{j+1}',\ldots,G_k',G_{k+1},\ldots,G_n)[f] \xRightarrow{a}_\diamond \\ \quad (G_1''',\ldots,G_j''',G_{j+1}',\ldots,G_k',G_{k+1},\ldots,G_n)[f]\end{array}$$

and, using Lemma 1 j times, we get

$$\begin{array}{l}(G_1''',\ldots,G_j''',G_{j+1}',\ldots,G_k',G_{k+1},\ldots,G_n)[f] \xRightarrow{a}_\diamond \\ \quad (G_1',\ldots,G_k',G_{k+1},\ldots,G_n)[f]\end{array}$$

which yields the matching transition. The other half of the proof is similar and therefore omitted. □

In the remainder of the paper we give a number of examples, showing the usefulness of MPL with the notion of observational refinement for verification purposes.

5 A simple data link

Joachim Parrow showed in [12] that $\mathcal{CCS}$ versions of the buffer B and the data link, P_J, incorporating a faulty medium, satisfy

$$P_J \approx B \tag{1}$$

We show that

$$P \trianglelefteq B \tag{2}$$

where P is an MPL specification, including as an implementation P_J. Thus our proof of (2) subsumes that of 1

B *must* be able to take in information through an a-action, whereupon it *must* release it using a $\overline{b}$-action. After that we still have a 1-place buffer. This is reflected in the specification

$$B \Longrightarrow a_{\Box}.\overline{b}_{\Box}.B$$

The data link is described through S, M, and R (Figure 2):

$$P = S \parallel M \parallel R$$

The *sender* S *must* admit information via an a-action, emitting it on $\overline{c}$, thus communicating with M, then becoming a sender. S *must* also be able to receive the acknowledgment sent by R, again becoming a sender. Since one may not know whether the medium used is faulty, a fault-handling capability *must* be included. With the exception handling bracketed we get

$$\begin{array}{rcl} S & \Longrightarrow & a_{\Box}.\overline{c}_{\Box}.S' \\ S' & \Longrightarrow & f_{\Box}.S\,[+d_{\Box}.\overline{c}_{\Box}.S'] \end{array}$$

All implementations of the *medium* M *must* admit pulses on c and emit them again with an $\overline{e}$-action. We might thus specify a perfect medium M_p as

$$\begin{array}{rcl} M_p & \Longrightarrow & c_{\Box}.M_p' \\ M_p' & \Longrightarrow & \overline{e}_{\Box}.M_p \end{array}$$

Since we do not want to rule out the possibility of a faulty but honest M we introduce an *optional* exception-handling part, which responds to a failure by communicating with S by a $\overline{d}$-action. But then the medium *must* be ready for a retransmission at c. Again putting the exception-handler in brackets, we get

$$\begin{array}{rcl} M & \Longrightarrow & c_{\Box}.M' \\ M' & \Longrightarrow & \overline{e}_{\Box}.M\left[+\overline{d}_{\Diamond}.M\right] \end{array}$$

The perfect medium of course implements the possibly faulty medium, i.e. we have $M_p \trianglelefteq M$, for it is easily seen that $\{(M_p, M), (M_p', M')\}$ is an observational refinement.

The *receiver* R has no exception parts. It *must* admit pulses on e, emit them on $\overline{b}$ and send an $\overline{f}$-acknowledgment:

$$R \Longrightarrow e_{\Box}.\overline{b}_{\Box}.\overline{f}_{\Box}.R$$

We show (2) by exhibiting an observational refinement. The behaviour of the specifications is given by the transition diagram in Figure 3 (Observe the τ-loop, indicating the possibility of divergence. The states are defined by

$$\begin{array}{rcl}
Q_1 & = & (\overline{c}_{\Box}.S' \parallel M \parallel R) \\
Q_2 & = & (S' \parallel M' \parallel R) \\
Q_3 & = & (S' \parallel M \parallel \overline{b}_{\Box}.\overline{f}_{\Box}.R) \\
Q_4 & = & (S' \parallel M \parallel \overline{f}_{\Box}.R)
\end{array}$$

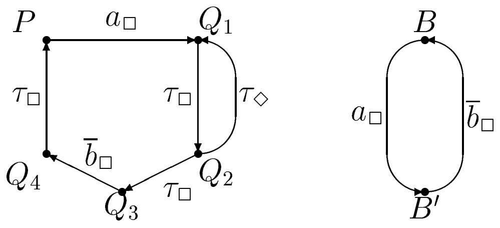

Figure 3: Data link and 1-place buffer

The sought observational refinement is

$$\{(P,B),(Q_1,B'),(Q_2,B'),(Q_3,B'),(Q_4,B)\}$$

Take a look at (Q_1, B'), for instance. Suppose that $Q_1 \xrightarrow{\tau}_{\Diamond} Q_2$ (the only $\Diamond$ -transition possible here). Then $\phi(\tau) = \epsilon$, $B' \stackrel{\epsilon}{\Longrightarrow}_{\Diamond} B'$, and we have (Q_2, B') in the relation. And $B' \xrightarrow{\overline{b}}_{\Box} B$ can be matched by $Q_1 \xrightarrow{\tau^2\overline{b}}{}^{*}_{\Box} Q_4$, implying that $Q_1 \stackrel{\overline{b}}{\Longrightarrow}_{\Box} Q_4$ with (Q_4, B) again in the relation.

Returning briefly to the 'perfect' medium, we saw that $M_p \lhd M$. Since $\parallel$ is a τ–insensitive static construct, it preserves $\unlhd$. Thus, we may conclude that $S \parallel M_p \parallel R \unlhd B$, which is what we would expect: a data link with a perfect medium should also be an implementation of the buffer. And since both $S \parallel M_p \parallel R$ and B possess only $\Box$ -transitions, they can be regarded as process expressions and $\unlhd$ as $\approx$, thus also establishing an observational equivalence. (Along the same lines, we can prove Joachim Parrow's original result for a data link with a faulty medium.)

6 A restartable system

In his Ph.D. thesis [14], K.V.S. Prasad considers a restartable system consisting of an *essential system* Q - here for simplicity the 1-place buffer - a *demon* D making Q crash, and an *exception handler* H evoked after the crash, setting up a new essential system with the state of the old one. (Figure 4.)

The *demon* D is simply a process which *may* interrupt the essential system through an $\overline{f}$-action, then returning to its demonic behaviour. In MPL, this is modelled as

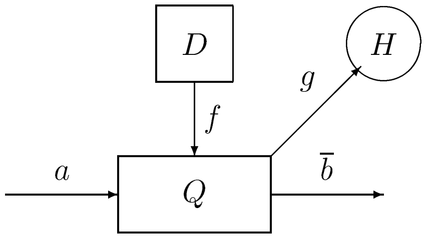

Figure 4: A simple restartable system

$$D \Longrightarrow \overline{f}_{\diamond}.D$$

The *essential system* Q contains the 1-place-buffer and *possibly* a part that, after the f-crash, stops the buffer behaviour, evokes the handler and kills the crashed system:

$$Q \Longrightarrow a_{\Box}.\overline{b}_{\Box}.Q\,[+f_{\diamond}.\overline{g}_{\Box}.Nil]$$

As before, the exception-handling part is put in brackets. Notice that although we do not require the essential system to respond to the demon, *if* so, it *must* evoke the handler.

The *handler* H *must*, after a g-signal from Q, produce a new Q together with a new handler:

$$H \Longrightarrow g_{\Box}.(Q \parallel H)$$

The entire system, $SF = (Q \parallel H \parallel D)$ observationally refines our 1-place buffer, i.e.

$$SF \trianglelefteq B \tag{3}$$

Denoting $Nil \parallel Nil \cdots \parallel Nil$ n times by Nil^n we can name the states of SF as

$$\begin{array}{rcl} S_k & = & (Nil^k \parallel D \parallel Q \parallel H) \\ S_k^b & = & (Nil^k \parallel D \parallel \overline{b}_{\Box}.Q \parallel H) \\ S_k' & = & (Nil^k \parallel D \parallel \overline{g}_{\Box}.Nil \parallel g_{\Box}.(Q \parallel H)) \end{array}$$

which intuitively corresponds to

S_k	The system after k crashes, ready to receive on a
S_k^b	The system after k crashes, ready to release on $\overline{b}$
S_k'	The system right after the $k+1$th crash

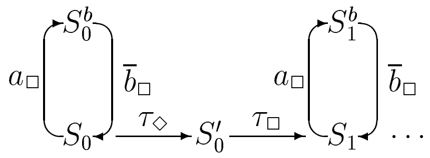

Figure 5: Modal transition diagram for restartable system

We get the modal transition diagram in Figure 5.

Divergence may occur here because of the possibility of always 'moving to the right' in the diagram (the system then keeps crashing and recovering). From Figures 5 and 3 we see that (3) holds, from the observational refinement

$$\{(S_k, B) \mid k \geq 0\} \cup \{(S_k^b, B') \mid k \geq 0\} \cup \{(S_k', B) \mid k \geq 0\}$$

where $SF = S_0$.

To see this, suppose that $S_k \xrightarrow{a}_\Diamond S_k^b$. Then $B \overset{a}{\rightarrow}{}^*_\Diamond B'$, implying that $B \overset{a}{\Longrightarrow}_\Diamond B'$, and (S_k^b, B') is in the relation. Or, if $S_k \xrightarrow{\tau}_\Diamond S_k'$ we have that $B \overset{\epsilon}{\rightarrow}{}^*_\Diamond B$, implying that $B \overset{\epsilon}{\Longrightarrow}_\Diamond B$ and (S_k', B) is in the relation. If $B \xrightarrow{a}_\Box B'$ we have $S_k \xrightarrow{a}_\Box S_k^b$ and then $S_k \overset{a}{\Longrightarrow}_\Box S_k^b$ with (S_k^b, B') in the relation.

The fact that (3) holds — together with the precongruence property of $\trianglelefteq$ with respect to $\|$ — shows that both a fault-tolerant system with a *nil*-demon and one where the demon is ignored implement the buffer.

7 A generalized data link

As our third example we generalize the data link specification, showing that our generalized data link observationally refines a general buffer. We only consider the modal transition diagrams, since MPL formulae here do not seem to aid in the understanding.

In [8] a general buffer which may accumulate a number of pulses before releasing them was described (Figure 6). The specification says nothing about the capacity of the buffer; a buffer may even not allow any information to enter to begin with. But if a buffer has information pulses in it, it *must* be able to release them. We assume again that we let in information on an a-action and release it with a b-action. The state of a buffer is characterized by the number of pulses it is holding; call a buffer with n pulses B_n.

The data link specification can be extended similarly as shown in the following:

The state of the *sender* S is characterized by the number of acknowledgments still to be received, m, and the number of pulses still to be sent to the medium, n. If a sender in state $S_{m,n}$ receives a pulse at a, it *may* let it in, becoming $S_{m,n+1}$. Hence for $m \geq 0$ we demand $S_{m,n} \xrightarrow{a}_\Diamond S_{m+1,n+1}$. Upon receiving a fault-report from the medium on d, S *must* respond to this. Thus $S_{m,n} \xrightarrow{d}_\Box S_{m,n+1}$. When S receives an acknowledgement at f it *must* respond by becoming $S_{m-1,n}$. Hence $S_{m,n} \xrightarrow{f}_\Box S_{m-1,n}$. Finally, the sender *must* be able to communicate with the medium via c, thus $S_{m,n} \xrightarrow{\overline{c}}_\Box S_{m,n-1}$, leading to the modal transition diagram in Figure 7.

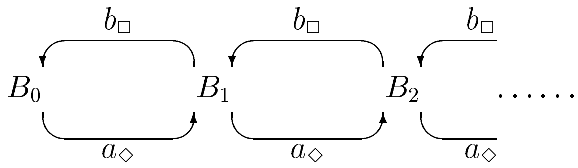

Figure 6: Specification of pulse buffer

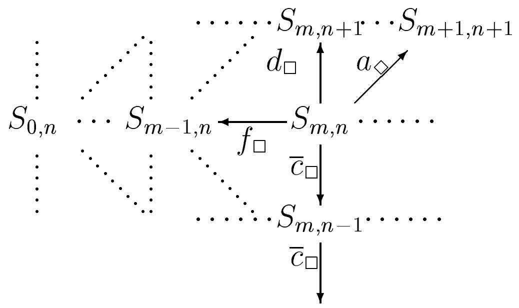

Figure 7: A generalized sender

The state of the *medium* M is characterized by the number of pulses held, l. M *must* be able to take in and release at least one pulse since information already in S must reach the receiver. This yields $M_0 \stackrel{c}{\rightarrow}_\Box M_1$ and $M_l \stackrel{c}{\rightarrow}_\Diamond M_{l+1}$ for $l > 0$. For all $l > 0$ we have $M_l \stackrel{\overline{e}}{\rightarrow}_\Box M_{l-1}$. Again it is not essential that the medium is faulty, so $M_l \stackrel{\overline{d}}{\rightarrow}_\Diamond M_{l-1}$ for $l > 0$. We here get the diagram of Figure 8.

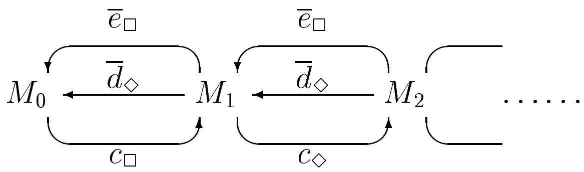

Figure 8: A generalized medium.

The *receiver* R is characterized by the number of acknowledgments, p, yet to be sent, and by the number of pulses, q, yet to be emitted. As for M, R *must* accept at least one pulse on e, since we must allow information in the system to leave. Thus $R_{p,0} \stackrel{e}{\rightarrow}_\Box R_{p+1,1}$. For $q > 0$ we only let $R_{p,q} \stackrel{e}{\rightarrow}_\Diamond R_{p+1,q+1}$. Any information in the R must always be able to leave, so we require $R_{p,q} \stackrel{\overline{b}}{\rightarrow}_\Box R_{p,q-1}$ for $q > 0$. And R must be able to send the acknowledgment, so we have $R_{p,q} \stackrel{\overline{f}}{\rightarrow}_\Box R_{p-1,q}$. This gives us Figure 9.

Now we have described the transition capabilities for generalized components in a generalized data

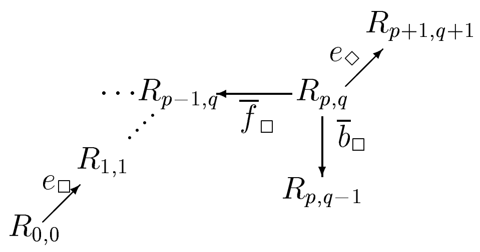

Figure 9: A generalized receiver

link. Next step is to show that a buffer B_k containing k information pulses is an abstract description of a data link containing k information pulses:

$$(S \parallel M \parallel R)_k \trianglelefteq B_k \tag{4}$$

But, what is a data link containing k pulses ? Answer: It is a system where the components altogether contain k pulses. In fact, one can see the data link as consisting of three communicating buffers. We thus only consider compositions $S_{m,n} \parallel M_l \parallel R_{p,q}$ where $k = n + l + q$.

And an observational refinement establishing (4) is the following relation:

$$\mathbf{R} = \{(S_{m,n} \parallel M_l \parallel R_{p,q}, B_k) \mid k = n + l + q\}$$

To see this, let us examine the transitions of the generalized data link using the operational rules of MPL.

One type of transitions gives information about information entering the data link. For $n > 0$ we get

$$S_{m,n} \parallel M_l \parallel R_{p,q} \xrightarrow{a}_\diamond S_{m+1,n+1} \parallel M_l \parallel R_{p,q}$$

This transition says that we by performing an a may bring one more pulse into the data link by placing it in the sender component.

Other transitions state that information can always leave the system when the receiver is non-empty ($q > 0$):

$$S_{m,n} \parallel M_l \parallel R_{p,q} \xrightarrow{\overline{b}}_\square S_{m,n} \parallel M_l \parallel R_{p,q-1}$$

All other transitions are unobservable. The sender and medium must be able to communicate a pulse if there are no pulses in the medium and pulses in the sender ($l = 0, n > 0$).

$$S_{m,n} \parallel M_0 \parallel R_{p,q} \xrightarrow{\tau}_\square S_{m,n-1} \parallel M_1 \parallel R_{p,q}$$

For $l > 0, n > 0$ we get

$$S_{m,n} \parallel M_l \parallel R_{p,q} \xrightarrow{\tau}_\Diamond S_{m,n-1} \parallel M_{l+1} \parallel R_{p,q}$$

Similarly, if there are no pulses in receiver and the medium is non-empty $(l > 0, q = 0)$, they must communicate a pulse:

$$S_{m,n} \parallel M_l \parallel R_{p,0} \xrightarrow{\tau}_\Box S_{m,n} \parallel M_{l-1} \parallel R_{p,1}$$

For $l > 0, q > 0$ we get

$$S_{m,n} \parallel M_l \parallel R_{p,q} \xrightarrow{\tau}_\Diamond S_{m,n} \parallel M_{l-1} \parallel R_{p,q+1}$$

If the medium reports a failure this must be because it lost a pulse $(l > 0)$ and then an unobservable communication may take place:

$$S_{m,n} \parallel M_l \parallel R_{p,q} \xrightarrow{\tau}_\Diamond S_{m,n+1} \parallel M_{l-1} \parallel R_{p,q}$$

There must at any time be the possibility of the communication of an acknowledgment from the receiver to the sender, if there are any to send $(m > 0, p > 0)$:

$$S_{m,n} \parallel M_l \parallel R_{p,q} \xrightarrow{\tau}_\Box S_{m-1,n} \parallel M_l \parallel R_{p-1,q}$$

The *buffer transitions* are

$$B_k \xrightarrow{a}_\Diamond B_{k+1}$$

$$B_k \xrightarrow{\overline{b}}_\Box B_{k-1}$$

We now easily see why $\mathbf{R}$ is an observational refinement. For, with $k = n + l + q$ we have that $(S_{m,n} \parallel M_l \parallel R_{p,q}) \xrightarrow{a}_\Diamond (S_{m+1,n+1} \parallel M_l \parallel R_{p,q})$ can be matched directly by $B_k \xrightarrow{a}_\Diamond B_{k+1}$ and $((S_{m+1,n+1} \parallel M_l \parallel R_{p,q}), B_{k+1}) \in \mathbf{R}$. Else, if $(S_{m,n} \parallel M_l \parallel R_{p,q}) \xrightarrow{\tau}_\Diamond$, these unobservable transitions always preserve the size of $n+l+q$, as is easily seen, and can be matched by $B_k \overset{\epsilon}{\Longrightarrow}_\Diamond B_k$, the results still in $\mathbf{R}$.

Any $\overline{b}_\Box$-transition made by the buffer must be due to $k > 0$. This means that at least one of n, l and q is non-zero, and thus we can by a series of $\tau_\Box$-transitions (not changing $n + l + q$) and a final $\overline{b}_\Box$-transition match this such that $n + l + q$ also decreases by one.

Notice that we in the above never used the sizes of m and p. This shows that the acknowledgment capabilities of the sender and receiver were here not essential to the specification. We might alternatively have marked the transitions here as $\Diamond$ -transitions or even left them out; we would still get (4).

Actually, we also have that

$$\mathbf{R}^T = \{(B_k, (S_{m,n} \parallel M_l \parallel R_{p,q}) \mid k = n + l + q\}$$

is an observational refinement, meaning that the buffer also observationally refines the data link ! This ensures that no potential implementation of the buffer specification has been lost by the chosen decomposition: *any* buffer can be implemented as a composition of components satisfying the specifications of the sender, medium, and receiver considered here.

Moreover, the present result not only gives an alternative proof of the simple data link and buffer introduced in section 5.1; it also shows how even the data link not allowing any information to enter and the data link with a very faulty medium (i.e. one that always loses its pulse thus performs an infinite sequence of d-transitions) observationally implement a buffer.

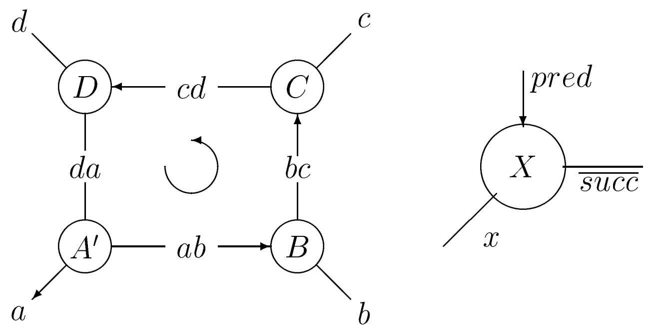

Figure 10: A simple scheduler and one of its typical cells

8 A simple scheduler

In this final section we reexamine the simple scheduler from [7] in the framework of MPL. The scheduler consists of 4 cyclic cells A, B, C and D (Figure 10).

A typical scheduler cell X (Figure 10) can be described in $\mathcal{CCS}$ by

$$\begin{array}{rcl} X & \Leftarrow & pred.X' \\ X' & \Leftarrow & x.X'' \\ X'' & \Leftarrow & \overline{succ}.X \end{array}$$

We obtain A, B, C and D by the following renamings:

$$\begin{array}{rcl} A & =^{\Delta} & X[da/pred, a/x, ab/succ] \\ B & =^{\Delta} & X[ab/pred, b/x, bc/succ] \\ C & =^{\Delta} & X[bc/pred, c/x, cd/succ] \\ D & =^{\Delta} & X[cd/pred, d/x, da/succ] \end{array}$$

The scheduler Sch is then defined as

$$Sch = (A' \| B \| C \| D)$$

We want to prove that Sch observationally refines the specification $Spec$:

$$Spec \Longrightarrow a_{\Box}.b_{\Box}.c_{\Box}.d_{\Box}.Spec$$

We could prove this fact directly by exhibiting an observational refinement containing the pair $(Sch, Spec)$. However, we prefer to give a compositional proof in order to demonstrate the use of MPL in that respect. Thus, we decompose Sch into the two subsystems $A' \| B$ and $D \| C$ as indicated in Figure 11.

The two subsystems will obviously communicate over the ports da and bc, and it is intuitively clear that the interaction between the two subsystems will consist of a simple alternation of bc

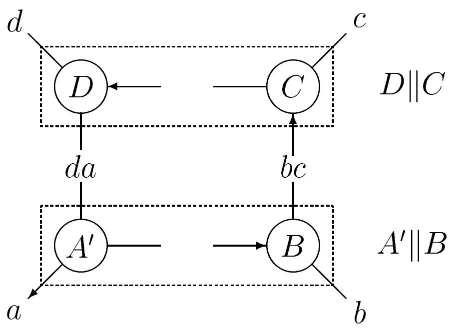

Figure 11: Decomposition of the scheduler

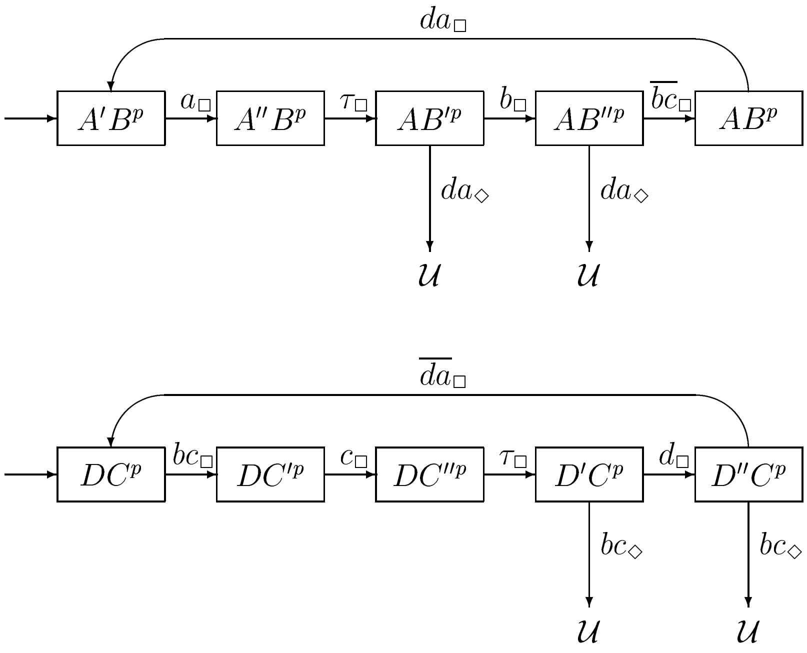

Figure 12: Transition diagrams of the scheduler subcomponents

and da, starting with bc. However, the interactions between the two subsystems are subject to certain restrictions. As a consequence not all of their potential behaviour will be accessible in the total system. In fact, one can show that approximately 50% of the potential behaviour of the two subsystems is never used. However, any subspecifications *equivalent* to the two subsystems must also cover this inaccessible behaviour and will therefore be unduly complicated.

The idea is now to use the looseness of MPL, so that only the accessible part of the subsystems are specified. In the present example it seems natural to suggest the following MPL specifications, represented directly as modal transition systems in Figure 12.

In these subspecifications the universal MPL formula $\mathcal{U}$ is introduced whenever an "area" is inaccessible. The transitions leading to these inaccessible areas (i.e. those led into by transitions marked $da_\Diamond$ and $bc_\Diamond$) are themselves inaccessible; this is reflected by the chosen modality ($\Diamond$).

To establish the correctness of the scheduler, we first prove the correctness of the subsystems with respect to the subspecifications, i.e.:

$$A'\|B \quad \trianglelefteq \quad A'B^p \tag{5}$$

$$D\|C \quad \trianglelefteq \quad DC^p \tag{6}$$

To conclude the correctness proof, we show that the subspecifications — when combined — entail the global specification, i.e.:

$$A'B^p\|DC^p \trianglelefteq Spec \tag{7}$$

Then, using the transitivity of $\trianglelefteq$ and the fact that it is preserved by $\|$, we conclude:

$$Sch = A'\|B\|C\|D \trianglelefteq Spec$$

9 Concluding remarks

We have in this paper shown the use and usefulness of a modal process logic incorporating static process constructs. Some points deserve to be stressed here.

- The MPL modalities $\Box$ and $\Diamond$ can be used in several different ways for expressing the looseness of specifications:
  - The 1-place buffer and restartable system specifications use the modalities for distinguishing between 'normal' and exceptional transitions, using $\Diamond$ for the latter.
  - The generalized buffer and data link specifications also use the modalities for emphasizing the generality of the specification. Here $\Diamond$ –transitions are the ones leading to states corresponding to instances of the specification exhibiting a more complicated behaviour.
  - The scheduler specification uses the $\Diamond$ –modalities to 'tone down' the transitions of its components that become inaccessible in the completed system.
- The incorporation of static constructs into MPL gives a natural way of specifying concurrent systems. Furthermore, the observational preorder $\trianglelefteq$ is a precongruence w.r.t. a large class of static constructs, facilitating a modular verification style.
- The verification technique used in this paper closely resembles that used in bisimulation proofs [14]. This should be seen in contrast to the verification technique employed by Sören Holmström in [2], in which he relies heavily on a number of established laws of a Hennessy–Milner logic extended with static process constructs and least fixed points. Of course one might contemplate basing the proofs presented in our paper on axiomatizations of $\triangleleft$ and $\trianglelefteq$.

References

[1] Hennessy, M. and R.Milner: *Algebraic Laws for Nondeterminism and Concurrency*, J.Assoc.Comput.Mach. 32, pp. 137–161, 1985.

[2] Holmström, S.: *Reasoning about CCS agents using Hennessy–Milner logic extended with fixed points*, Workshop on Parallel Systems, Stirling, 1988.

[3] Hüttel, Hans: *Operational and Denotational Properties of a Modal Process Logic*, Master's Thesis, Department of Mathematics and Computer Science, Aalborg University Centre, 1988.

[4] Larsen, Kim G. : *A Context-Dependent Bisimulation Between Processes*, Ph.D. Thesis, University of Edinburgh, 1986.

[5] Larsen, Kim G.: *Proof Systems for Hennessy-Milner Logic with Recursion*, in: *CAAP 88*, LNCS 299, Springer-Verlag New York/Berlin 1988. (Extended version to appear in Theoretical Computer Science, North-Holland).

[6] Larsen, Kim and Robin Milner: *Verifying a Protocol Using Relativized Bisimulation*, in Proceedings of ICALP'87, LNCS 267, Springer-Verlag New York/Berlin 1987.

[7] Larsen, Kim and Bent Thomsen: *Compositional Proofs by Partial Specification of Processes*, in Proceedings of Mathematical Foundations of Computer Science 1988, LNCS 324, Springer-Verlag New York/Berlin 1988.

[8] Larsen, Kim. G and Bent Thomsen: *A Modal Process Logic*, in Proceedings of Third Annual symposium on Logic in Computer Science, Edinburgh, 1988.

[9] Milner, Robin: *A Calculus of Communicating Systems*, LNCS 92, Springer-Verlag New York/Berlin 1980.

[10] Milner, Robin: *Calculi for Synchrony and Asynchrony*, Theoretical Computer Science 25, pp. 267-310, North-Holland 1983.

[11] D.Park: *Concurrency and automata on infinite sequences*, in Proceedings og 5th GI Conference, LNCS 104, Springer-Verlag New York/Berlin 1981.

[12] Parrow, Joachim: *Fairness Properties in Process Algebra*, Ph.D. Thesis, Dept. of Computer Systems, Uppsala University, Uppsala 1985.

[13] Plotkin, Gordon: *A Structural Approach to Operational Semantics*, Tech. Rep., DAIMI FN-19, Computer Science Department, Aarhus University 1981.

[14] Prasad, K.V.S: *Combinators and Bisimulation Proofs for Restartable Systems*, Ph.D. Thesis, Dept. of Computer Science, University of Edinburgh 1987.