Technical Report

UCAM-CL-TR-264
ISSN 1476-2986

Number 264

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

CCS with environmental guards

Juanito Camilleri

August 1992

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

CCS with Environmental Guards

Juanito Camilleri

Department of Computer Studies
University of Malta
University Heights
Msida
Malta G.C.

Computer Laboratory
University of Cambridge
Pembroke Street
Cambridge
England CB2 3QG

Abstract

This paper investigates an extension of Milner's CCS with agents guarded by propositions on the environment. The agent $g \gg E$, pronounced *E in an environment of which g holds*, depends on the set of actions the environment is ready to perform. This dependency is realised by an operational semantics in which transitions carry ready-sets (of the environment) as well as the normal action symbols from CCS. A notion of strong bisimulation is defined on guarded agents via this semantics. It is a congruence and satisfies new equational laws (including a new expansion law) which are shown to be complete for finite guarded agents. The laws are conservative over agents of traditional CCS. The guarding operator $\gg$ provides a dynamic, local, and clean syntactic means of expressing the behaviour of an agent depending on circumstance; it is more expressive than the *unless* operator presented in [Cam91] and the priority choice operator presented in [Cam90] and [CaW91], and yields a much simpler expansion theorem.

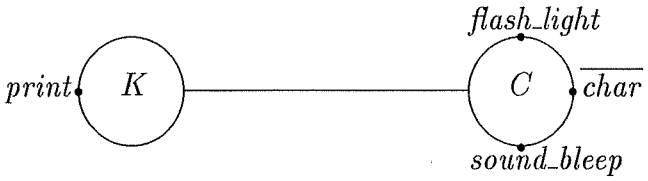
1 Introduction

This paper augments Milner's CCS by introducing agents that are *positively* and *negatively* constrained by guards which specify the environmental circumstances in which the agents can behave. A positive constraint on an agent makes its behaviour depend on the environment's ability to perform some action. A negative constraint makes an agent's behaviour depend on the environment's inability to perform some action. In other words, guards are propositions which can restrict the behaviour of an agent to environments that can perform certain actions and not certain others.

Guarding is both *dynamic* and *local*. It is dynamic in the sense that the environmental constraints on an agent may change as execution proceeds. It is local in the sense that the components of a system may behave independently subject to their local constraints.

We will show how the guarding operator provides a syntactic means of expressing a priority structure on actions, that it can encode a form of multi-way synchronisation, and that it can be specialised to several useful priority and conditional operators including *priority choice* [Cam90] [CaW91] and the *unless* operator presented in [Cam91]. The work presented here can be adopted as a formal semantic foundation for existing programming language constructs that embody the notion of priority between actions [Cam89] [Cam90] and hints at ways in which the expressiveness of such constructs can be augmented.

Consider, for example, a reactive system *Sys* (see Fig.1). A keyboard is busy-waiting for the environment to issue the command *print*, meanwhile the printer-controller cannot



$$
\begin{array}{lll}
Sys & \stackrel{def}{=} & (keyboard \mid controller)\backslash a \\
keyboard & \stackrel{def}{=} & print.\overline{a}.\mathbf{0} \; + \; [\neg \mathsf{print}] \gg \tau.keyboard \\
controller & \stackrel{def}{=} & [\mathsf{on\_line} \wedge \neg \mathsf{out\_of\_paper} \wedge \neg \mathsf{jammed\_paper}] \gg a.\overline{char}.\mathbf{0} \; + \\
 & & [\mathsf{out\_of\_paper}] \gg flash\_light.controller \; + \\
 & & [\neg \mathsf{on\_line} \vee \mathsf{jammed\_paper}] \gg sound\_bleep.controller
\end{array}
$$

Figure 1: A simple reactive system.

issue a "print-character" signal: if the printer is out of paper the printer-controller flashes a light, if the printer is not on-line or the paper is jammed then the controller sounds a bleep. As soon as the environment can perform $\overline{print}$, the keyboard is required to stop waiting immediately. If the printer is on-line, has not run out of paper and is not jammed then the controller is expected to handshake with the keyboard and then issue a "print-character" signal to the printer.

This scenario can be specified very naturally using guarded agents. Suppose the agent $g \gg E$ behaves as E in an environment of which proposition g holds, then the system Sys can then be defined as shown in Fig.1.

2 CCS with environmental guards

Suppose we simply augment CCS with terms of the form $g \gg E$ where g is a proposition on the possible actions of the environment and E is any expression in the augmented language. We would then allow mutually dependent expressions such as

$$((a.\mathbf{0} \; + \; [\neg \overline{\mathsf{a}}] \gg b.\mathbf{0}) \mid (\overline{b}.\mathbf{0} \; + \; [\neg \mathsf{b}] \gg \overline{a}.\mathbf{0})) \setminus \{a, b\}. \tag{1}$$

The left operand of the parallel composition can perform an a-action or, if the environment cannot perform $\overline{a}$ the operand can perform b. On the other hand, the right operand of the composition can perform $\overline{b}$ or, if the environment cannot perform a b action, the operand can perform $\overline{a}$. Can this composition perform a τ action? One can argue both ways. Partly to avoid deciding such questions, and partly to allow a simple expansion law (see Fig.5), we restrict the syntax of agent expressions. We will break the symmetry in CCS that exists between actions a and their complements $\overline{a}$, and distinguish between i-actions and o-actions:

- an i-action is a named action in $\mathcal{A}$ with typical elements $a, b, c, \ldots$,

- an o-action is a co-named action in $\overline{\mathcal{A}}$ with typical elements $\overline{a}, \overline{b}, \overline{c}, \ldots$,
- a *silent action* is represented by τ.

We shall denote by Act the set of *actions* $\mathcal{A} \cup \overline{\mathcal{A}} \cup \{\tau\}$ ranged over by α. We shall understand the operation $\overline{(\ )}$ to act so $a \mapsto \overline{a}$, $\overline{a} \mapsto a$. The operation $\overline{(\ )}$ acts as identity on τ. For every $a \in \mathcal{A}$ we define a *basic guard* $\mathsf{a} : \mathcal{P}(\overline{\mathcal{A}}) \to \mathit{bool}$ where

$$\mathsf{a} \stackrel{def}{=} \lambda R.\ \overline{a} \in R$$

Let $\mathcal{B}$ be the set of basic guards ranged over by $\boldsymbol{\alpha}$. A guard g is an assertion of the language $\mathcal{G}$ defined by:

$$g ::= \boldsymbol{\alpha} \mid \mathsf{T} \mid \mathsf{F} \mid g \vee g \mid g \wedge g \mid \neg g$$

For a set $R \subseteq \overline{\mathcal{A}}$ and an assertion $g \in \mathcal{G}$, we say R *satisfies* g, when $R \models g$ is true subject to the definition

$$\begin{array}{lcl} R \models \boldsymbol{\alpha} & = & \boldsymbol{\alpha}\ R \\ R \models \mathsf{T} & = & \mathit{true} \\ R \models \mathsf{F} & = & \mathit{false} \\ R \models g_0 \vee g_1 & = & R \models g_0 \ \mathit{or}\ R \models g_1 \\ R \models g_0 \wedge g_1 & = & R \models g_0 \ \mathit{and}\ R \models g_1 \\ R \models \neg g & = & \mathit{not}(R \models g) \end{array}$$

where *and*, *or* and *not* are boolean conjunction, disjunction and negation respectively.

CCS augmented with guarded agents is called CCS$^{\gg}$. Let $\mathcal{E}^{\gg}$ be the set of agent expressions with typical elements $E, F \ldots$, and let $\mathcal{G}^{\gg}$ be the set of guarded expressions with typical elements $G, H \ldots$. Let X range over the set of agent variables and P, Q and N range over the set of closed terms $\mathcal{P}^{\gg}$. The syntax of CCS$^{\gg}$ expressions is defined as follows:

$$G ::= X \mid \mathbf{0} \mid a.E \mid \tau.E \mid G \setminus L \mid G[f] \mid G + G \mid \mathit{fix}(X = G) \mid g \gg G$$
$$E ::= X \mid \mathbf{0} \mid \alpha.E \mid E \setminus L \mid E[f] \mid E + E \mid E|E \mid \mathit{fix}(X = E) \mid G$$

We assume that the relabelling function f in $E[f]$ is injective and maps $\mathcal{A}$ to $\mathcal{A}$ and $\overline{\mathcal{A}}$ to $\overline{\mathcal{A}}$. Furthermore, let L in $E \setminus L$ be a subset of $\mathcal{A}$. Finally, let recursion be guarded.

We intend propositions of the language $\mathcal{G}$ to constrain the behaviour of agents. Note that guards g in expressions of the form $g \gg G$ can only constrain G by making it dependent on the possible o-action behaviour of the environment. Guarded agents, however, can only perform i-actions or the silent action. Hence problems due to mutual dependency as illustrated by (1) are avoided.

3 Operational semantics

The behaviour of CCS$^{\gg}$ agents is formalised by a transition relation $\vdash_R E \stackrel{\alpha}{\longrightarrow} E'$ to be understood as meaning: in an environment which is ready to perform the actions R, the

re (i) $g \gg G$ re $\emptyset$	re (ii) $\mathbf{0}$ re $\emptyset$
re (iii) $\tau.E$ re $\emptyset$	re (iv) $a.E$ re $\emptyset$
re (v) $\overline{a}.E$ re $\{\overline{a}\}$	
re (vi) $\dfrac{E_0 \text{ re } R_0 \quad E_1 \text{ re } R_1}{E_0 + E_1 \text{ re } R_0 \cup R_1}$	re (vii) $\dfrac{E \text{ re } R}{E \setminus L \text{ re } R - \overline{L}}$ $(L \subseteq \mathcal{A})$
re (viii) $\dfrac{E[fix(X = E)/X] \text{ re } R}{fix(X = E) \text{ re } R}$	re (ix) $\dfrac{E \text{ re } R}{E[f] \text{ re } f(R)}$
re (x) $\dfrac{E \text{ re } R_0 \quad F \text{ re } R_1}{E \mid F \text{ re } R_0 \cup R_1}$	

Figure 2: The definition of the ready-function.

agent E can perform an action α to become the agent E'. The ready-set R is a subset of $\overline{\mathcal{A}}$. The transition relation will be defined in terms of a *ready-function.* Given an agent E, the ready-function yields the set containing those actions in $\overline{\mathcal{A}}$ that E can do next in any environment. The ready-function is defined inductively in Fig.2.

The rules defining the transition relation are presented in Fig.3. Since the other rules are similar to those of CCS, we comment only on the rules for guarded commands and composition. Rule Guard states that $g \gg G$ can perform α in an environment that is ready to perform R, provided G can perform α and the ready-set R satisfies the guard g. Now consider the rule Com_c for composition under the assumptions that E re R_0 and F re R_1. The rule takes account of the fact that the assumption that the environment of $E|F$ is ready to do R amounts to the assumption on E that its environment is ready with $R \cup R_1$, and similarly that F's environment is ready with $R \cup R_0$. If under these assumptions E and F can perform complementary actions then their composition can synchronise. The rule Com_c has a pleasing symmetry, though note that by virtue of Prop.3.1 the requirements of Com_c, Com_a and Com_b can be relaxed when they involve o-actions. Consequently, these rules could be replaced by two rules taking account of the fact that transitions associated with o-actions do not depend on ready-sets of the environment.

Proposition 3.1 *For terms E, E' and all $R, S \subseteq \overline{\mathcal{A}}$, $\alpha \in \overline{\mathcal{A}}$,*

$$\vdash_R E \xrightarrow{\alpha} E' \iff \vdash_S E \xrightarrow{\alpha} E'.$$

Proposition 3.2 *The ready-relation* re *is a function in the sense that E re R and E re R' implies $R = R'$. Moreover*

$$E \,\mathsf{re}\, S \iff S = \{\alpha \in \overline{\mathcal{A}} \,:\, \forall R \exists E'. \vdash_R E \xrightarrow{\alpha} E'\}$$

Notation: In future we will use $\mathsf{re}(E)$ for the ready-set of E—justified because the ready-relation is a function.

CCS$^{\gg}$ essentially extends CCS; once we restrict to unguarded agents, the transition relations agree but for the extra decoration of ready-sets on the relations.

Proposition 3.3 *For an unguarded term E, $E \xrightarrow{\alpha} E'$ in* CCS *iff*
$\vdash_R E \xrightarrow{\alpha} E'$ *in* CCS$^{\gg}$ *for all $R \subseteq \overline{\mathcal{A}}$ such that, if $\alpha \in \mathcal{A}$ then $\overline{\alpha} \in R$.*

As mentioned earlier, assertions of the language $\mathcal{G}$ can be used to impose both positive or negative constraints on the environment. A positive constraint on an agent makes its behaviour depend on the environment's ability to perform some o-action, while a negative constraint makes an agent's behaviour depend on the environment's inability to perform some o-action. Any assertion in $\mathcal{G}$ can be reduced to disjunctions of conjunctions of positive and negative constraints:

Proposition 3.4 *Any $g \in \mathcal{G}$ can be reduced to the* disjunctive normal form

$$\bigvee_{i \in I} \bigwedge_{j \in J_i} \boldsymbol{\beta}_{ij}$$

where $\boldsymbol{\beta}_{ij} \equiv \boldsymbol{\alpha}$ or $\boldsymbol{\beta}_{ij} \equiv \neg\boldsymbol{\alpha}$ for some $\boldsymbol{\alpha} \in \mathcal{B}$, for all i in I and j in J_i. The disjunctive normal form denotes T *when J_i are empty for all $i \in I$, it denotes* F *when for all $i \in I$ there exists some j_0, j_1 in J_i and $\boldsymbol{\alpha} \in \mathcal{B}$ such that $\boldsymbol{\beta}_{ij_0} \equiv \boldsymbol{\alpha}$ and $\boldsymbol{\beta}_{ij_1} \equiv \neg\boldsymbol{\alpha}$.*

$$\mathsf{Pre}_a \quad \vdash_R a.E \xrightarrow{a} E \ \text{ if } \overline{a} \in R$$

$$\mathsf{Pre}_c \quad \vdash_R \overline{a}.E \xrightarrow{\overline{a}} E \qquad\qquad \mathsf{Pre}_d \quad \vdash_R \tau.E \xrightarrow{\tau} E$$

$$\mathsf{Guard} \quad \frac{\vdash_R G \xrightarrow{\alpha} E'}{\vdash_R g \gg G \xrightarrow{\alpha} E'} \ R \models g$$

$$\mathsf{Sum}_a \quad \frac{\vdash_R E \xrightarrow{\alpha} E'}{\vdash_R E + F \xrightarrow{\alpha} E'} \qquad\qquad \mathsf{Sum}_b \quad \frac{\vdash_R F \xrightarrow{\alpha} F'}{\vdash_R E + F \xrightarrow{\alpha} F'}$$

$$\mathsf{Res} \quad \frac{\vdash_{R-\overline{L}} E \xrightarrow{\alpha} E'}{\vdash_R E \setminus L \xrightarrow{\alpha} E' \setminus L} \ (\alpha \notin \overline{L}) \qquad\qquad \mathsf{Rel} \quad \frac{\vdash_R E \xrightarrow{\alpha} E'}{\vdash_{f(R)} E[f] \xrightarrow{f(\alpha)} E'[f]}$$

$$\mathsf{Com}_a \quad \frac{\vdash_{R \cup R_1} E \xrightarrow{\alpha} E' \quad F \ \mathsf{re} \ R_1}{\vdash_R E|F \xrightarrow{\alpha} E'|F} \ (\alpha \in \mathcal{A} \Longrightarrow \overline{\alpha} \in R)$$

$$\mathsf{Com}_b \quad \frac{\vdash_{R \cup R_0} F \xrightarrow{\alpha} F' \quad E \ \mathsf{re} \ R_0}{\vdash_R E|F \xrightarrow{\alpha} E|F'} \ (\alpha \in \mathcal{A} \Longrightarrow \overline{\alpha} \in R)$$

$$\mathsf{Com}_c \quad \frac{\vdash_{R \cup R_1} E \xrightarrow{\alpha} E' \quad E \ \mathsf{re} \ R_0 \quad \vdash_{R \cup R_0} F \xrightarrow{\overline{\alpha}} F' \quad F \ \mathsf{re} \ R_1}{\vdash_R E|F \xrightarrow{\tau} E'|F'} \ (\alpha \neq \tau)$$

$$\mathsf{Rec} \quad \frac{\vdash_R E[fix(X = E)/X] \xrightarrow{\alpha} E'}{\vdash_R fix(X = E) \xrightarrow{\alpha} E'}$$

Figure 3: The definition of the transition relation.

One can simplify the syntax of guards further by noticing the pleasing relationship between the disjunction of guards and the nondeterministic choice of guarded agents:

Proposition 3.5 *For all ready-sets $R \subseteq \overline{\mathcal{A}}$, actions $\alpha \in Act$, guarded-agents $G \in \mathcal{G}^{\gg}$,*

$$\vdash_R (\bigvee_{i\in I} \bigwedge_{j\in J_i} \beta_{ij}) \gg G \xrightarrow{\alpha} E \iff \vdash_R \sum_{i\in I} (\bigwedge_{j\in J_i} \beta_{ij}) \gg G \xrightarrow{\alpha} E$$

Therefore, any guarded agent in $\mathcal{G}^{\gg}$ can be re-expressed as an agent whose subterms may only contain guards which are conjunctions of positive and negative constraints. To allow a clear expression of the equational theory in the next section, we represent a conjunction of positive and negative constraints as two respective sets of actions.

Notation: Henceforth, we will consider only those assertions $g \in \mathcal{G}$ of the form $\bigwedge_{j\in J} \beta_j$ where $\beta_j \equiv \alpha$ or $\beta_j \equiv \neg\alpha$ for some $\alpha \in \mathcal{B}$. Such assertions are said to be in *normal form* and will be denoted by a tuple (g^+, g^-) where

$$g^+ = \{\overline{a} \ : \ \mathsf{a} \equiv \beta_j \ \wedge \ j \in J\}$$

and

$$g^- = \{\overline{a} \ : \ \neg\mathsf{a} \equiv \beta_j \ \wedge \ j \in J\}$$

We will use $g_0 \cup g_1$ to denote $(g_0^+ \cup g_1^+, g_0^- \cup g_1^-)$ and $g_0 \subseteq g_1$ when $g_0^+ \subseteq g_1^+$ and $g_0^- \subseteq g_1^-$.

If a proposition g is in normal form and R is the ready-set of the environment, then $R \models g$ holds when g^+ is a subset of R and none of the actions in g^- are elements of R.

Proposition 3.6 *For all propositions $g \in \mathcal{G}$ which are in normal form:*

$$R \models (g^+, g^-) \iff g^+ \subseteq R \ \wedge \ g^- \cap R = \emptyset$$

If a guard g_0 implies g_1 and R satisfies g_0 then R satisfies g_1:

Proposition 3.7 *For all propositions $g_0, g_1 \in \mathcal{G}$ in normal form, for all $R \subseteq \overline{\mathcal{A}}$:*

$$R \models g_0 \ \wedge \ g_1 \subseteq g_0 \Longrightarrow R \models g_1$$

4 Strong bisimulation

We take a generalisation of Milner's strong bisimulation as our central equivalence between agents.

Definition 4.1 A relation $\mathcal{Q} \subseteq \mathcal{P}^{\gg} \times \mathcal{P}^{\gg}$ is a strong bisimulation with respect to the operational semantics, if $(P, Q) \in \mathcal{Q}$ implies, for all $\alpha \in Act$ and for all $R \subseteq \overline{\mathcal{A}}$,

1. whenever $\vdash_R P \xrightarrow{\alpha} P'$ then, for some Q', $\vdash_R Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{Q}$,
2. whenever $\vdash_R Q \xrightarrow{\alpha} Q'$ then, for some P', $\vdash_R P \xrightarrow{\alpha} P'$ and $(P', Q') \in \mathcal{Q}$.

Definition 4.2 Let the relation $\sim$ on agents be defined by:

$$P \sim Q \text{ iff } (P, Q) \in \mathcal{Q} \text{ for some strong bisimulation } \mathcal{Q}.$$

As emphasised by Park, it follows by basic fixed-point theory that $\sim$ is an equivalence relation and the largest strong bisimulation, and as such it is amenable to the proof technique: to show $P \sim Q$ it is sufficient to exhibit a strong bisimulation containing (P, Q).

This technique is used to show that equivalence is a congruence, i.e., it is substitutive under the constructs of CCS$^{\gg}$ as well as under recursive definition. The proof follows the standard lines of [Mil89], though with this more intricate operational semantics, it is necessary to check that the strong equivalence of two agents implies that they have the same ready-sets (this is needed in proving the congruence of $\sim$ with respect to parallel composition):

Lemma 4.1 *If* $P \sim Q$ *then* $\mathsf{re}(P) = \mathsf{re}(Q)$.

Proposition 4.2 *Let* $P_1 \sim P_2$, *then*

(1) $\alpha.P_1 \sim \alpha.P_2$	*(3)* $g \gg P_1 \sim g \gg P_2$	*(5)* $P_1 \mid Q \sim P_2 \mid Q$
(2) $P_1 + Q \sim P_2 + Q$	*(4)* $P_1 \setminus L \sim P_2 \setminus L$	*(6)* $P_1[f] \sim P_2[f]$

Thus strong equivalence is preserved by prefixing, summation, guarding, parallel composition, restriction and relabelling. We remark that, due to the syntactic restriction on guarded terms, P_1 and P_2 in statement *(3)* must be terms in $\mathcal{G}^{\gg}$, otherwise $g \gg P_1$, $g \gg P_2$ etc. would not be terms in $\mathcal{E}^{\gg}$. For example, let $E \equiv (\overline{a}.\mathbf{0} + b.\mathbf{0}) \setminus a$ and $F \equiv b.\mathbf{0}$—i.e., suppose that F is an element of $\mathcal{G}^{\gg}$ and E is not. Although $E \sim F$, $g \gg E \not\sim g \gg F$ for $g \in \mathcal{G}$.

Hitherto, strong equivalence has been defined only for closed expressions. To remedy this, we extend the definition of $\sim$ to open terms as follows. Let *substitution* σ map term variables to closed terms such that $E\sigma$ represents the closed term resulting from the substitution of all free variables X in E by $\sigma(X)$.

Definition 4.3 Let E and F be agent terms, possibly with free variables. Then define $E \sim F$ to hold exactly when for all substitutions σ, $E\sigma \sim F\sigma$.

It is a simple matter to extend Prop.4.2 to open terms. Prop.4.3 shows, moreover, that recursion preserves strong equivalence.

Proposition 4.3 *If* $E \sim F$ *then* $fix(X = E) \sim fix(X = F)$

Theorem 4.4 $\sim$ *is a congruence with respect to the operators of* CCS$^{\gg}$.

Finally we note that the extension of CCS with guarded agents does not lead to any new identifications between closed terms of CCS:

Proposition 4.5 *Two unguarded terms* E, E' *are strongly equivalent in the sense of Milner iff* $E \sim E'$ *in* CCS$^{\gg}$.

This means that the equational laws of the next section are conservative over CCS terms: terms of CCS will only be provably equal iff they formerly were so in CCS.

5 Equational laws

We now present a set of equational laws which are complete with respect to finite CCS$^{\gg}$ terms (A term is *finite* if it contains no variables). First, the usual rules of equational reasoning (reflexivity, symmetry, transitivity and Liebnitz' rule, *viz.*"substitution of equals for equals") hold as $\sim$ is a congruence. Further rules are presented in Fig.4. These consist of Milner's laws for strong equivalence together with new laws for guarded agents, notably **G1**-**G8**, **R4** and **L4**.

G1 states that if $\overline{a} \in g^-$ then $g \gg a.P$ behaves as $\mathbf{0}$.

G2 states that an a-prefix also acts as a positive constraint $\overline{a}$.

G3 deals with the case when T guards an agent.

G4 deals with the case when F guards an agent.

G5 states that guarding the agent $\mathbf{0}$ does not change its behaviour.

G6 shows that a guard can distribute over nondeterministic choice.

G7 states that nested guards can be re-expressed as a conjunction.

G8 states that if g_0 implies g_1 then a choice between G guarded by g_0 and G guarded by g_1 behaves as G guarded by g_1.

Law **R4** asserts the distribution of guarding over relabelling. Note that if g is a guard then $f(g)$ denotes the relabelling of all atomic propositions in g—e.g. if f maps a to b and c to d then

$$f((\{\overline{c}\},\{\overline{a}\}) \gg G) = (\{\overline{d}\},\{\overline{b}\}) \gg G$$

Finally, one can view a restriction $E \setminus L$ as an agent E whose behaviour evolves in an environment that cannot perform actions in $\overline{L}$. Therefore we expect $(g \gg G)\setminus L$ to behave as $\mathbf{0}$ if some action in $\overline{L}$ appears in g^+. On the other hand, if $\overline{L}$ and g^+ are disjoint, then since restriction acts as a form of negative constraint, one can expect $(g \gg G) \setminus L$ to behave as $(g^+, g^- - \overline{L}) \gg (G \setminus L)$ (see law **L4**).

Excluding **Rec1** and **Rec2**, the laws of Fig.4 with the addition of the expansion law **EXP** (Fig.5) are complete for finite terms. The expansion law operates on terms in *guarded normal form.*

Definition 5.1 A term P is said to be in *guarded normal form*:

$$\sum_{m \in M} g_m \gg \alpha_m.P_m$$

- if for all $m \in M$, P_m is in guarded normal form, and
- for all $m \in M$, if $\alpha_m \in \mathcal{A}$ then $\overline{\alpha}_m \notin g_m^-$, and $g_m^+ \cap g_m^- = \emptyset$.

We write "$(\emptyset,\emptyset) \gg \overline{a}.P$" for "$\overline{a}.P$".

A1 $P + Q = Q + P$ **A2** $P + P = P$

A3 $P + (Q + R) = (P + Q) + R$ **A4** $P + \mathbf{0} = P$

G1 $g \gg a.P = \mathbf{0}$ if $\overline{a} \in g^-$ **G2** $g \gg a.P = (g^+ - \{\overline{a}\}, g^-) \gg a.P$

G3 $(\emptyset, \emptyset) \gg G = G$ **G4** $g \gg G = \mathbf{0}$ if $g^+ \cap g^- \neq \emptyset$

G5 $g \gg \mathbf{0} = \mathbf{0}$ **G6** $(g \gg G) + (g \gg H) = g \gg (G + H)$

G7 $g_0 \gg (g_1 \gg G) = (g_0 \cup g_1) \gg G$

G8 $(g_0 \gg G) + (g_1 \gg G) = g_1 \gg G$ if $g_1 \subseteq g_0$

R1 $\mathbf{0}[f] = \mathbf{0}$ **R2** $(P + Q)[f] = P[f] + Q[f]$

R3 $(\alpha.P)[f] = f(\alpha).(P[f])$ **R4** $(g \gg G)[f] = f(g) \gg G[f]$

L1 $\mathbf{0} \setminus L = \mathbf{0}$

L2 $(\alpha.P) \setminus L = \begin{cases} \mathbf{0} & \text{if } \alpha \in L \cup \overline{L} \\ \alpha.(P \setminus L) & \text{otherwise} \end{cases}$

L3 $(P + Q) \setminus L = (P \setminus L) + (Q \setminus L)$

L4 $(g \gg G) \setminus L = \begin{cases} \mathbf{0} & \text{if } \overline{L} \cap g^+ \neq \emptyset \\ (g^+, g^- - \overline{L}) \gg (G \setminus L) & \text{otherwise} \end{cases}$

Rec1 $fix(X = E) = E[fix(X = E)/X]$

Rec2 $F = fix(X = E)$ if $F = E[F/X]$

Figure 4: Equational laws satisfied by $\sim$.

Let $P \equiv \sum_{m\in M} g_m \gg \alpha_m.P_m$ and $Q \equiv \sum_{n\in N} h_n \gg \beta_n.Q_n$

be two terms in guarded normal form. Then

$$\begin{aligned} P|Q \;=\; & \sum_{m\in M} \{(g_m^+ - \mathsf{re}(Q), g_m^-) \gg \alpha_m.(P_m|Q) : \mathsf{re}(Q) \cap g_m^- = \emptyset\} \;+ \\ & \sum_{n\in N} \{(h_n^+ - \mathsf{re}(P), h_n^-) \gg \beta_n.(P|Q_n) : \mathsf{re}(P) \cap h_n^- = \emptyset\} \;+ \\ & \sum_{\overline{\beta_n}=\alpha_m\in\mathcal{A}} \{(g_m^+ - \mathsf{re}(Q), g_m^-) \gg \tau.(P_m|Q_n) : \mathsf{re}(Q) \cap g_m^- = \emptyset\} \;+ \\ & \sum_{\overline{\alpha_m}=\beta_n\in\mathcal{A}} \{(h_n^+ - \mathsf{re}(P), h_n^-) \gg \tau.(P_m|Q_n) : \mathsf{re}(P) \cap h_n^- = \emptyset\} \end{aligned}$$

Figure 5: The expansion law **EXP**.

Note that any prefix component $\alpha_m.P_m$ of a term in guarded normal form is associated with a transition that the guarded term can make. When the indexing set M is empty, we have the **0** agent in guarded normal form. When E is in guarded normal form, the prefix $\alpha.E$ is in guarded normal form if the indexing set M is singleton and g_m is $(\emptyset, \emptyset)$ for $m \in M$. The summation, composition, restriction and relabelling of terms in guarded normal form can be reduced to a term in guarded normal form using the laws in Fig.4 and the expansion law in Fig.5. In the case of unguarded expressions the expansion law reduces to the one from CCS.

Notation The sum $\sum_{i\in I} E_i$ is indexed by a finite ordered set I. Because summation is commutative and associative with respect to $\sim$, we do not care about the precise order on the indexing set. We shall understand $\sum_{P(G_i)} G_i$, where P is a predicate, as an abbreviation for the sum $\sum_{i\in I'} G_i$ indexed by $I' = \{i \in I \mid P(G_i)\}$.

Theorem 5.1 *(Soundness)*
The equational laws of Fig.4 and Fig.5 are sound (i.e., for each law $P = Q$ it is true that $P \sim Q$). Any equation deduced by equational reasoning from these laws is valid as long as both sides of the equation are terms in $\mathcal{E}^{\gg}$.

Proof *(Soundness of Expansion Theorem)*

Suppose $P \equiv \sum_{m\in M} g_m \gg \alpha_m.P_m$ and $Q \equiv \sum_{n\in N} h_n \gg \beta_n.Q_n$,

and abbreviate the right-hand side of the expansion law in Fig.5 by *exp*. To establish $P|Q \sim exp$ it is sufficient to show

$$\vdash_R P|Q \xrightarrow{\mu} E \;\Leftrightarrow\; \vdash_R exp \xrightarrow{\mu} E$$

for all $R \subseteq \overline{\mathcal{A}}$, actions μ and agents E.

($\Rightarrow$) There are several ways in which a transition $\vdash_R P|Q \xrightarrow{\mu} E$ can be derived; either by rule Com_a, Com_b or Com_c. Consider only the latter case; the proof for the others is similar. In this case $\mu = \tau$ and

$$\vdash_{R\cup R_1} P \xrightarrow{\alpha} P' \text{ and } \vdash_{R\cup R_0} Q \xrightarrow{\overline{\alpha}} Q', \text{ such that } \mathsf{re}(P) = R_0 \text{ and } \mathsf{re}(Q) = R_1.$$

Assume $\alpha = a$ say in $\mathcal{A}$ (the argument when $\overline{a} \in \mathcal{A}$ is symmetric). Then from the form of Q we must have $\overline{a} = \beta_{n_0}$ and $Q' \equiv Q_{n_0}$ for some $n_0 \in N$. From the form of P, we see $a = a_{m_0}$ and $P' \equiv P_{m_0}$ for some $m_0 \in M$ such that:

$$\vdash_{R\cup R_1} g_{m_0} \gg \alpha_{m_0}.P_{m_0} \xrightarrow{a} P_{m_0}$$

The rule for guarding ensures that in order for this transition to be derivable $R\cup R_1 \models g_{m_0}$ must hold. By Prop.3.6,

$$g^+_{m_0} \subseteq R \cup R_1 \tag{2}$$

and

$$g^-_{m_0} \cap (R \cup R_1) = \emptyset. \tag{3}$$

Therefore

$$g^-_{m_0} \cap R = \emptyset \tag{4}$$

and

$$g^-_{m_0} \cap R_1 = \emptyset. \tag{5}$$

Consider now the sum

$$\sum_{\overline{\beta}_n = \alpha_m \in \mathcal{A}} \{(g^+_m - \mathsf{re}(Q), g^-_m) \gg \tau.(P_m \mid Q_n) \ : \ \mathsf{re}(Q) \cap g^-_m = \emptyset\}$$

occurring in the expression exp. Since $\mathsf{re}(Q) = R_1$ then from (5) we have that

$$(g^+_{m_0} - R_1, g^-_{m_0}) \gg \tau.(P_{m_0} \mid Q_{n_0})$$

is a summand in exp. From (2),

$$(g^+_{m_0} - R_1) \subseteq R \tag{6}$$

From (4) and (6), and Prop.3.6, $R \models (g^+_{m_0} - R_1, g^-_{m_0})$.
Consequently, $\vdash_R exp \xrightarrow{\tau} (P'|Q')$—i.e., the original transition of $(P|Q)$ is matched by one of exp.

($\Rightarrow$) One also requires that any transition of exp can be matched by one of $(P \mid Q)$. For example, one transition is:

$$\vdash_R exp \xrightarrow{\tau} (P_{m_0} \mid Q_{n_0})$$

where $\overline{\alpha}_{m_0} = \beta_{n_0} = \overline{a}$ say in $\mathcal{A}$. For this transition to be derivable one must necessarily have that

$$\vdash_R \sum_{\overline{\alpha}_m = \beta_n \in \mathcal{A}} \{(h^+_n - \mathsf{re}(P), h^-_n) \gg \tau.(P_m \mid Q_n) \ : \ \mathsf{re}(P) \cap h^-_n = \emptyset\} \xrightarrow{\tau} (P_{m_0} \mid Q_{n_0}) \tag{7}$$

From the rules of sum and guarding, however, this is only possible if

$$R \models (h^+_{n_0} - \mathsf{re}(P), h^-_{n_0}). \tag{8}$$

Assume that $\mathsf{re}(P) = R_0$ and $\mathsf{re}(Q) = R_1$. From Prop.3.6 and (8)

$$h^+_{n_0} - R_0 \subseteq R \tag{9}$$

and

$$h^-_{n_0} \cap R = \emptyset. \tag{10}$$

From (9),

$$h^+_{n_0} \subseteq R \cup R_0. \tag{11}$$

From (7),

$$R_0 \cap h^-_{n_0} = \emptyset. \tag{12}$$

Therefore from (10) and (12),

$$h^-_{n_0} \cap (R \cup R_0) = \emptyset. \tag{13}$$

By Prop.3.6, (11) and (13),

$$R \cup R_0 \models h_{n_0}$$

It follows from the guarding rule that

$$\vdash_{R \cup R_0} h_{n_0} \gg \beta_{n_0}.Q_{n_0} \xrightarrow{a} Q_{n_0}.$$

Hence $\vdash_{R \cup R_0} Q \xrightarrow{a} Q_{n_0}$ and as $\overline{a}$ is an o-action, $\vdash_{R \cup R_1} P \xrightarrow{\overline{a}} P_{m_0}$. Thus, the original transition of exp has been matched by one of $P \mid Q$. The other transitions are shown to match in a similar fashion.

∎

Theorem 5.2 *(Soundness and Completeness)*
For finite terms P, Q in $\mathcal{E}^{\gg}$, $P \sim Q$ if and only if $P = Q$.

Proof

Soundness follows from Theorem 5.1. Conversely, suppose $P \sim Q$. One can argue without loss of generality that P and Q are in guarded normal form. Therefore, we show by induction on the maximum depth of P and Q that $P \sim Q \Rightarrow P = Q$ where P and Q are in guarded form.

The base case of the induction when the depth of P and Q is 0 is established easily as then $P \equiv Q \equiv \mathbf{0}$ and $P = Q$ follows from **A2**. Otherwise assume that, P and Q have the following form:

$$P \equiv \sum_{m \in M} g_m \gg \alpha_m.P_m \quad \text{and} \quad Q \equiv \sum_{n \in N} h_n \gg \beta_n.Q_n$$

We will show that $Q = Q + (g_{m_0} \gg \alpha_{m_0}.P_{m_0})$ for all $m_0 \in M$. It then follows that $Q + P = Q$ and by a symmetric argument that $P + Q = P$. Hence $P = Q$. Suppose $g_{m_0} \gg \alpha_{m_0}.P_{m_0}$ is a summand of P for some $m_0 \in M$, we consider

$$\vdash_R P \xrightarrow{\alpha} P'$$

where $\alpha = \alpha_{m_0}$ and $P' \equiv P_{m_0}$ when $R = g^+_{m_0}$, and then when $R = \overline{\mathcal{A}} - g^-_{m_0}$. Since $P \sim Q$ and P, Q and their subterms are in strong guarded form, then there exists $n_0 \in N$, and a term Q' such that

$$\vdash_R Q \xrightarrow{\alpha} Q', \quad P' \sim Q', \quad \alpha = \beta_{n_0}, \quad Q' \equiv Q_{n_0} \quad \text{and} \quad R \models h_{n_0}.$$

By Prop.3.6, when $R = g^+_{m_0}$ we can deduce that $h^+_{n_0} \subseteq g^+_{m_0}$ and when $R = \overline{\mathcal{A}} - g^-_{m_0}$ we can deduce that $h^-_{n_0} \subseteq g^-_{m_0}$. Therefore $h_{n_0} \subseteq g_{m_0}$. By law **G8**,

$$Q = Q + (g_{m_0} \gg \beta_{n_0}.Q_{n_0}).$$

Since P' and Q' are subterms of P and Q, inductively, $P' = Q'$, and moreover, since $\alpha_{m_0} = \beta_{n_0}$, then

$$Q = Q + (g_{m_0} \gg \alpha_{m_0}.P_{m_0}).$$

This can be shown for all $m_0 \in M_0$. Therefore $Q = Q + P$. A converse argument shows that $P = P + Q$. Therefore $P = Q$. ∎

5.1 Applying the equational laws

Recall the reactive system Sys defined in the introduction. The system can be re-expressed as follows:

$$\begin{aligned}
Sys &\stackrel{def}{=} (keyboard \mid controller)\backslash a \\
keyboard &\stackrel{def}{=} print.\overline{a}.\mathbf{0} + (\emptyset, \{\overline{print}\}) \gg \tau.keyboard \\
controller &\stackrel{def}{=} (\{\overline{on_line}\}, \{\overline{out_of_paper}, \overline{jammed_paper}\}) \gg a.\overline{char}.\mathbf{0} + \\
&\quad (\{\overline{out_of_paper}\}, \emptyset) \gg flash_light.controller + \\
&\quad (\emptyset, \{\overline{on_line}\}) \gg sound_bleep.controller + \\
&\quad (\{\overline{jammed_paper}\}, \emptyset) \gg sound_bleep.controller
\end{aligned}$$

Applying **EXP**, **L4**, **L3** and **L2** yields:

$$\begin{aligned}
&(\emptyset, \{\overline{print}\}) \gg \tau.Sys + \\
&(\{\overline{out_of_paper}\}, \emptyset) \gg flash_light.Sys + \\
&(\emptyset, \{\overline{on_line}\}) \gg sound_bleep.Sys + \\
&(\{\overline{jammed_paper}\}, \emptyset) \gg sound_bleep.Sys + \\
&(\emptyset, \emptyset) \gg print.((\overline{a}.\mathbf{0} \mid controller)\backslash a)
\end{aligned}$$

$$
\begin{array}{lcl}
Meal & \stackrel{def}{=} & (Food \mid D_0 \mid D_1 \mid D_2 \mid Butler) \setminus \{gong, eat\} \\
D_0 & \stackrel{def}{=} & gong.(\neg\mathsf{gong} \gg port.\mathbf{0}) \; + \; \neg\mathsf{gong} \gg eat.D_0 \\
D_1 & \stackrel{def}{=} & gong.(\neg\mathsf{gong} \gg port.\mathbf{0}) \; + \; \neg\mathsf{gong} \gg eat.D_1 \\
D_2 & \stackrel{def}{=} & gong.(\neg\mathsf{gong} \gg port.\mathbf{0}) \; + \; \neg\mathsf{gong} \gg eat.D_2 \\
Butler & \stackrel{def}{=} & timeup.\overline{gong}.\overline{gong}.\overline{gong}.\mathbf{0} \\
Food & \stackrel{def}{=} & \overline{eat}.Food
\end{array}
$$

Figure 6: The dons' meal.

Further applications of **EXP**, **L1**, **L2**, **L4** and **G3** yields *Spec* defined as follows:

$$
\begin{array}{l}
(\emptyset, \{\overline{print}\}) \gg \tau.Sys \; + \\
(\{\overline{out\_of\_paper}\}, \emptyset) \gg flash\_light.Sys \; + \\
(\emptyset, \{\overline{on\_line}\}) \gg sound\_bleep.Sys \; + \\
(\{\overline{jammed\_paper}\}, \emptyset) \gg sound\_bleep.Sys \; + \\
print.(\{\overline{on\_line}\}, \{\overline{out\_of\_paper}, \overline{jammed\_paper}\}) \gg \tau.\mathbf{0}
\end{array}
$$

Therefore, *Sys* is strongly bisimilar to *Spec* which can be re-expressed as follows:

$$
\begin{array}{l}
[\neg\mathsf{print}] \gg \tau.Sys \; + \\
[\mathsf{out\_of\_paper}] \gg flash\_light.Sys \; + \\
[\neg\mathsf{on\_line} \; \vee \; \mathsf{jammed\_paper}] \gg sound\_bleep.Sys \; + \\
print.[\mathsf{on\_line} \; \wedge \; \neg\mathsf{out\_of\_paper} \; \wedge \; \neg\mathsf{jammed\_paper}] \gg \tau.\mathbf{0}
\end{array}
$$

6 The dining-dons example

Three Cambridge dons are sharing a limitless source of food. A butler waits patiently while the meal proceeds. When the butler notices that time is up, the dons are required to stop eating immediately, and the butler strikes a gong three times to declare the meal 'over'. The meal is defined in Fig.6. Before *timeup* occurs:

- Any don can help himself or herself to food.
- The food never runs out and the butler waits patiently.

After *timeup* occurs, the dons cannot drink port before three gong-strikes. In fact, one can prove (see [Cam90]) that the behaviour can only proceed as follows:

$$\tau.\tau.\tau.port.port.port.\mathbf{0}$$

Informally, suppose *timeup* occurs. This can only be followed by a $gong/\overline{gong}$ interaction between the butler and one of the dons—for example don D_0. After this first interaction D_0 is in the state:

$$\neg\mathsf{gong} \gg port.\mathbf{0}$$

while the butler is in state:

$$\overline{gong}.\overline{gong}.\mathbf{0}$$

Since the butler is still prefixed by a $\overline{gong}$ action, D_0 is prevented from drinking port while D_1 and D_2 are prevented from eating. Once again, only a $gong/\overline{gong}$ interaction may take place, this time between the butler and either don D_1 or don D_2. Suppose the butler interacts with don D_2, then the state of the butler becomes $\overline{gong}.\mathbf{0}$, and don D_2 like D_0 has state $\neg\mathsf{gong} \gg port.\mathbf{0}$. The remaining $\overline{gong}$ prefix of the butler prevents D_0 and D_2 from drinking port until the final $gong/\overline{gong}$ interaction takes place between the butler and D_1. This done, the butler stops and as there are no $\overline{gong}$ prefixes left in their environment, the dons are now free to have port.

This example illustrates the use of guarding to encode a form of multi-way synchronisation; notice that all the dons stop eating as soon as *timeup* occurs. The dons represent agents that share some resource. The butler represents a sensor. When the sensor detects an interrupt, the agents stop their normal behaviour. The sensor then ensures that each agent commences some routine to recover from the interrupt. Guarding prevents one of the agents to proceed with recovery before the sensor has initialised recovery in the other agents.

7 Related Work

This work arose as a continuation of [Cam89] which presented an operational semantics of the PRIALT construct in occam. The guarding operation presented here is more rudimentary than the priority choice operator *prisum* of [Cam90] and [CaW91]. Informally, G_0 *prisum* G_1, written as $G_0 \pitchfork G_1$, can be expressed as $G_0 + (\emptyset, A) \gg G_1$ where A is the acceptance set[1] of G_0. On the other hand, there is no way of expressing the behaviour of a simple guarded agent $(\emptyset, g^-) \gg G$ using the priority choice operator. In fact, terms of the form $(\emptyset, g^-) \gg G$ coincide with the conditional agents expressed by the *unless* operator of [Cam91]. It is clear that *unless* can only impose negative constraints on an agent and therefore the operator is less expressive than the guarding operator presented here.

Both *unless* and *prisum* have duals which are also special cases of guarding. One can represent an operator G *provided* g^+ as $(g^+, \emptyset) \gg G$. Like *unless*, *provided* can be used to enforce synchronisation strategies. Sometimes, however, one needs the happy marriage of both these operators realised by the guarding operator presented here. The dual of the priority choice operator is rather peculiar. It can be expressed as $G_0 + (A, \emptyset) \gg G_1$ where A is the acceptance set of G_0. This agent behaves nondeterministically like G_0 or G_1 in an environment that allows all the i-actions of G_0 to occur. None of the actions of G_1 can occur in an environment that refuses an i-action of G_0.

[1]The acceptance set of an agent is the set of o-actions whose complement is the set of i-actions which the agent can perform next.

We know of no other successful attempt to both semantically define and, with this as a basis, provide a complete proof system for a CCS-like language with agents guarded by both positive and negative constraints. We contribute a new set of laws for guarded agents, complete for finite agents, as part of a well-rounded and, we believe, convincing theory.

There have been other attempts to give a semantic basis to reasoning about priority [BBK85] [Bar89] [ClH90] [BeK90] [Gro90] [GeL90] [SmS90] [Tof90] [CaW91] [Cam91] (see [Cam90] for more details). These attempts fall under three main categories:

- those that associate priority with choice [Cam89] [Bar89] [SmS90] [Tof90] [CaW91],
- those that associate priority with events [ClH90] [BeK90] [GeL90], and
- those that express priority through environmental constraints [Cam90] [Cam91].

The guarding operator presented here falls under the last category. As stated in the introduction, guarding:

- is a primitive and expressive means of imposing local, dynamic, positive and negative constraints on agents without complicating the underlying theory of CCS unreasonably,
- can be used to encode a form of multiway synchronisation,
- can reflect syntactically a priority structure of actions in each state, and
- lends itself to implementation—it is closely related to existing priority constructs in programming languages [Cam89] and hints at how the expressiveness of such constructs may be extended.

Future work, should look at how to extend the theory of weak bisimulation to cater for CCS agents with environmental guards. This is not expected to be trivial [Jen92]. It is evident that guarding and other specialised priority constructs can be used to impose intricate scheduling strategies including ones that are fair [CoS84], however, this relationship requires further study. Finally, the implementation of the concurrency workbench which caters for priority choice [Jen91], can be adapted to handle the more expressive and simpler guarding operator presented here. The simple equational theory developed in this paper makes its automated use with tools such as PAM [Lin91] feasible. An extension of the current work on embedding CCS in the proof assistant HOL [Nes91] to handle agents with environmental guards is also possible.

8 Acknowledgements

I am grateful to Richard Boulton and Glynn Winskel for some helpful comments. Many thanks to Glynn Winskel for funding a visit to Aarhus, and to Mike Gordon for supporting my stay at Cambridge where most of the work presented here was done. Finally, I would like to thank the University of Malta for granting me a sabbatical for research.

References

[Bar89] G. Barrett. The Semantics of Priority and Fairness in **occam**, April 1989. Proc. MFPS 5, New Orleans, USA.

[BBK85] J.C.M. Baeten, J.A. Bergstra and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. Technical Report CS-R8503, Center for Mathematics and Comp. Sci, Amsterdam, February 1985.

[BeK90] E. Best and M. Koutny. Partial order semantics of priority systems. Technical Report 6/90, Institute of Computer Science, University of Hildesheim, June 1990.

[Bri86] E. Brinksma. A tutorial on LOTOS. In M Diaz, editor, *Protocol Specification, testing, and verification, V*, pages 171–194. Elsevier Science Publishers B.V., 1986.

[Cam89] Juanito Camilleri. An operational semantics for **occam**. *International Journal of Parallel Programming*, 18(5), October 1989.

[Cam90] Juanito Camilleri. *Priority in Process Calculi.* Ph.D thesis (October 1990). Technical Report 227, Computer Laboratory, University of Cambridge.

[CaW91] Juanito Camilleri and Glynn Winskel. CCS with Priority Choice. Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science, July 1991. To appear in *Information and Computation.*

[Cam91] Juanito Camilleri. A Conditional Operator for CCS. In J.C.M. Baeten and J.F. Groote, editors, *ConCur'91*, LNCS 527, pages 142–156, Springer Verlag, August 1991.

[ClH90] R. Cleaveland and M. Hennessy. Priorities in process algebras. *Information and Computation*, Vol.87, Nos.1/2, July/August 1990.

[CoS84] G. Costa and C. Stirling. Weak and strong fairness in CCS. In Chytil M P and Koubek V, editors, *Mathematical Foundations of Computer Science*, LNCS 176, pages 245–264. Springer Verlag, 1984.

[GeL90] R. Gerber and I. Lee. CCSR: A Calculus for Communicating Shared Resources. In J.C.M. Baeten and J.W.Klop, editors, *ConCur'90*, LNCS 458, pages 263–277, Springer Verlag 1990.

[Gro90] Jan Friso Groote. Transition system specifications with negative premises. In J.C.M. Baeten and J.W.Klop, editors, *ConCur'90*, LNCS 458, pages 332–341, Springer Verlag 1990.

[inm84] inmos. **occam** *Programming Manual.* International Series in Computer Science. Prentice Hall, 1984

[Jen91] Claus Torp Jensen. The Concurrency Workbench with Priorities. In K. Larsen and A. Skou, editors, *CAV'91*, LNCS 575, pages 147–157, Springer Verlag, July 1991.

[Jen92] Claus Torp Jensen. Ph.D Thesis Proposal. University of Aarhus, Denmark, April 1992.

[Lin91] Huimin Lin. PAM: A Process Algebra Manipulator. In K. Larsen and A. Skou, editors, *CAV'91*, LNCS 575, pages 136–146, Springer Verlag, July 1991.

[Mil89] Robin Milner. *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, 1989.

[Nes91] Monica Nesi. Mechanizing a Proof by Induction of Process Algebra Specifications in Higher Order Logic. In K. Larsen and A. Skou, editors, *CAV'91*, LNCS 575, pages 288–298, Springer Verlag, July 1991.

[SmS90] S. Smolka and B. Steffen. Priority as Extremal Probability. In J.C.M.Baeten, J.W.Klop, editors, *ConCur'90*, LNCS 458, pages 456–466, Springer Verlag 1990.

[Tof90] C. Tofts. A Synchronous Calculus of Relative Frequency. In J.C.M.Baeten, J.W.Klop, editors, *ConCur'90*, LNCS 458, pages 467–480, Springer Verlag 1990.

[WiN] G. Winskel and M. Nielsen. Models for concurrency (to appear in S. Abramsky, D.M. Gabbay, T.S.E. Maibaum eds. *Handbook of Logic in Computer Science*).