Reasoning about Higher-Order Processes

Roberto M. Amadio[1]* and Mads Dam[2]**

[1] CNRS, BP 145, Sophia-Antipolis, F-06903, France, e-mail: amadio@cma.cma.fr
[2] SICS, Box 1263, S-164 28, Kista, Sweden, e-mail: mfd@sics.se

Abstract. We address the specification and verification problem for process calculi such as Chocs, CML and Facile where processes or functions are transmissible values. Our work takes place in the context of a static treatment of restriction and of a bisimulation-based semantics. As a paradigmatic and simple case we concentrate on (Plain) Chocs. We show that Chocs bisimulation can be characterized by an extension of Hennessy-Milner logic including a constructive implication, or function space constructor. This result is a non-trivial extension of the classical characterization result for labelled transition systems. In the second part of the paper we address the problem of developing a proof system for the verification of process specifications. Building on previous work for CCS we present a sound proof system for a Chocs sub-calculus not including restriction. We present two completeness results: one for the full specification language using an infinitary system, and one for a special class of so-called *well-described* specifications using a finitary system.

1 Introduction

In the last years there has been a rising interest in calculi and programming languages where *complex* data such as processes and functions are transmissible values [3, 5, 11, 13, 18]. At least two main motivations for these studies can be identified: (i) to generalize the functional model of computation to a parallel and/or concurrent framework, and (ii) to model the notion of code transmission which is relevant to the programming of distributed systems.

A key issue in these languages is the interaction between process transmission and the *static* scoping discipline for communication channels. Here we consider Thomsen's Plain Chocs. This is an extension of CCS where processes are transmissible values and the restriction operator is subject to a static scoping discipline.

A considerable effort has been put into the development of a bisimulation based semantics for this calculus (c.f. [18, 2, 14]). The *specification* of Plain Chocs processes (and processes in related calculi) is a much less developed topic. Two notable attempts in this direction are described in [17, 6]. These works are based on logics extracted from a domain theoretic interpretation of the calculus,

* Partially supported by ESPRIT BRA 6454 CONFER. Part of the work was done while visiting SICS.
** Partially supported by ESPRIT BRA 8130 LOMAPS.

following general ideas described in, e.g., [1]. This approach has been rather successful in the case of *dynamic* scoping. On the other hand it is not clear how to obtain a fully abstract denotational semantics of restriction in the case of *static* scoping (c.f. [12] for some typical problems). This motivates our shift towards an operational approach to the problem, along the lines of Hennessy and Milner [8].

What to specify? First, let us fix some notation for a process calculus of higher-order processes: $c!P.P'$ is the process which sends P along the channel c and then becomes P', $c?x.P$ is the process which performs an input along the channel c, and upon reception of some process Q becomes $[Q/x]P$. In $\nu c.P$ the restriction operator ν creates a new channel which will be local to the process P. Finally $+$ is the non-deterministic choice, $|$ is the parallel composition, and 0 is the nil process, with the usual CCS semantics [9]. We briefly refer to this calculus as Chocs, after [17].

Second, we should determine some requirements for our candidate specification logic. Roughly, we expect it to be an extension of Hennessy-Milner logic which characterizes some standard Chocs bisimulation.

Previous work on extending Hennessy-Milner logic to calculi including value and channel transmission (c.f. [4, 7, 10]) relies on the recurrent idea of introducing modalities that state properties of the transmitted values. For instance, one can specify that a process P can output the value 3 on channel c and then satisfy property ϕ by writing: $P : \langle c!3\rangle.\phi$.

This approach does not seem to scale up to process transmission. The naive idea of writing:

$$P : \langle c!\phi'\rangle.\phi \text{ if } P \overset{c!P'}{\rightarrow} Q \text{ and } P' : \phi' \text{ and } Q : \phi$$

does not take into account the fact that P' and Q might share local channels. For instance, consider the process $P = \nu a.c!(a?x.x).a!0.Q$. In this example the actions of the process transmitted on channel c and of the relative continuation are clearly inter-dependent. We did not find any satisfying way to express this dependency. An alternative is to express properties of processes like P above in terms of the effect the output has when P is put in a receiving environment. Since receiving environments are just abstractions this suggests a simple extension of Hennessy-Milner logic by means of a constructive implication, say $\Rightarrow$. We can now write specifications such as:

$$P : [c?](\phi \Rightarrow \psi) \qquad Q : \langle c!\rangle((\phi \Rightarrow \psi) \Rightarrow \gamma)$$

For inputs the interpretation is the expected one: P satisfies $[c?](\phi \Rightarrow \psi)$ if whenever P makes an input action on channel c and receives an input satisfying ϕ then the continuation will satisfy ψ. In a first approximation, the intuition for output is the following: Q satisfies $\langle c!\rangle((\phi \Rightarrow \psi) \Rightarrow \gamma)$ if it is possible for Q to output a process, say Q_1, along c such that in any receiving context, say $\lambda x.Q_2$, if $\lambda x.Q_2$ satisfies $\phi \Rightarrow \psi$, and if Q_3 is the continuation of Q after performing its output, then $([Q_1/x]Q_2) \mid Q_3$ satisfies γ. In general Q_1 and Q_3

may share channels local to Q, say $c_1, ..., c_n$, whose scope is *extruded* by the output communication. Note that in the specification we never need to speak about these extruded channels.

Operational Semantics. Reflecting this intuition, a labelled transition system is given to Plain Chocs that maps (closed) processes to (closed) process functionals, depending on the action performed. More precisely, a process Q rewrites by an input action to a process function $\lambda x.Q_2$ and by an output action to a process functional $\lambda f.\nu c_1...c_n.(fQ_1 \mid Q_3)$, where, as above, Q_1 is the transmitted process, Q_3 is the continuation, and $c_1, ..., c_n$ are shared local channels. The result of a communication is simply computed by applying the process functional to the process function:

$$(\lambda f.\nu c_1...c_n.(fQ_1 \mid Q_3))(\lambda x.Q_2) \equiv \nu c_1...c_n.([Q_1/x]Q_2 \mid Q_3)$$

The standard rules for substitution avoid clashes between local channels. Also note that in this formulation Plain Chocs actions coincide with standard CCS actions. Of course one has to pay a price for this, namely one has to lift the notion of bisimulation higher-order by introducing a suitable notion of exponentiation. Section 2 will show that this can be achieved in an elegant and simple way. It should be remarked that the resulting bisimulation coincides with the one considered in [2, 14], which in turn has been shown to be compatible with the π-calculus semantics.

Logical Characterization. Having found a suitable way to specify properties of Chocs processes we pursue our programme of relating logical equivalence to bisimulation equivalence. In the CCS case, this is achieved by means of a co-inductive view of bisimulation. Roughly, the bisimulation relation, say $\sim$, can be seen as the limit of a descending sequence of equivalence relations $\sim^k$. Equivalence in $\sim^k$ is then related to logical equivalence w.r.t. formulas having modal depth bound by k. In the higher-order case the task is complicated by the *contravariance* of the constructive implication in its first argument. This is discussed in more detail in section 2 once some notation has been introduced. We obtain a logical characterization of Chocs bisimulation modulo a technical lemma that relates the approximation $\sim^k$ to a *sharpened* approximation $\sim^k_\sharp$.

Towards a Proof System. As a second contribution, we address the problem of developing a sound and complete proof system to verify that a process meets (or realizes) a specification. We found a simple and clean solution for the restriction-free fragment of the calculus. The basic judgment $\Gamma \vdash P : \psi$ states that the process P realizes the specification ψ under the hypothesis Γ. Hypotheses state assumptions on the parameters of P. The system thus allows for reasoning about open processes. A rather rough completeness result for the system can be achieved by the introduction of an ω-rule and by the hypothesis that there is only a finite number of channels. To give a more accurate picture of the power of our logical system we also exhibit a finitary system which is complete on a particular collection of *well-described* specifications. Concerning the restriction operator it appears this may require considerable complication of the proof system as one has to represent the dependencies among functional variables and dynamically generated channels. We leave this problem for further investigation.

2 The Calculus and its Bisimulation Based Semantics

Language. The expressions of the language are classified in two kinds: channels and processes. Channels are variables and ranged over by $c, d, \ldots$. Actions have one of the forms τ, $c?$ or $c!$ and they are ranged over by $\alpha, \alpha', \ldots$. To each process is associated a unique order among the orders: 0 (processes), 1 (process functions), and 2 (process functionals). We use $x, x', \ldots$, $f, f' \ldots$, and $F, F' \ldots$, for variables of order $0, 1, 2$, respectively. We use $z, z' \ldots$ as generic variables. Open processes of order 0 are then generated by the following grammar:

$$P ::= 0 \,\Big|\, x \,\Big|\, fP \,\Big|\, P + P \,\Big|\, P \mid P \,\Big|\, \nu c.P \,\Big|\, c!P.P \,\Big|\, c?x.P$$

Whenever we write $P[z]$ we intend that z is the only variable that can be free in P and moreover we identify P with the function $\lambda z.P$. Thus alpha-conversion applies to identify $P[z]$ with $([z'/z]P)[z']$ whenever z' does not occur freely in $\lambda z.P$, and to identify, e.g., $\nu c.(P[z])$ with $(\nu c.P)[z]$. We also write $P(z)$ for an open process in which z is the only variable that can occur free. If z is free then $P(z)$ is identified with $P[z]$. If z is not free then $P(z)$ can ambiguously represent either a closed process or the constant function $\lambda z.P$. The context will allow us to disambiguate this situation.

Operational semantics. The labelled transition system is based on three kinds of judgments: $P \xrightarrow{\tau} Q$, $P \xrightarrow{c?} Q'[x]$, and $P \xrightarrow{c!} Q''[f]$, where P, Q are closed processes. We assume that sum and parallel composition are associative and commutative operators, and that restriction commutes with parallel composition according to the standard law $(\nu c.P) \mid P' = \nu c.(P \mid P')$ whenever c is not free in P'. Then it can be showed that whenever $P \xrightarrow{c!} Q''[f]$ in the transition system specified below then $Q''[f]$ has the form $\nu c_1.\cdots\nu c_n.(fP' \mid P'')[f]$. Finally, note that in the rule (!?) a *second-order substitution* is employed. That is, one replaces first $Q'[x]$ for f, and then the argument of f for x.

(!) $c!P'.P \xrightarrow{c!} (fP' \mid P)[f]$ (?) $c?x.P \xrightarrow{c?} P[x]$
(!?) if $P \xrightarrow{c!} P'[f]$ and $Q \xrightarrow{c?} Q'[x]$ then $P \mid Q \xrightarrow{\tau} [Q'[x]/f]P'[f]$
(+) if $P \xrightarrow{\alpha} P'$ then $P + Q \xrightarrow{\alpha} P'$ (|) if $P \xrightarrow{\alpha} P'$ then $P \mid Q \xrightarrow{\alpha} P' \mid Q$
(ν) if $P \xrightarrow{\alpha} P'$ and $\alpha \neq c!, c?$ then $\nu c.P \xrightarrow{\alpha} \nu c.P'$

Bisimulation. Let Pr_0 be the collection of closed processes, Pr_1 be the collection of $P[x]$ processes, and Pr_2 be the collection of $P[f]$ processes. Because of the input-output actions a notion of bisimulation over Pr_0 needs to be lifted to

Pr_1 and Pr_2. For this purpose the following general notion of *exponentiation* is introduced:

$$P[z]\ [S \Rightarrow S']\ P'[z] \text{ if } Q(w)\ S\ Q'(w) \text{ implies } [Q(w)/z]P\ S'\ [Q'(w)/z]P'$$

Given a relation S over Pr_0^2 and an action α we define the relations $S[\alpha]$ as follows, where $Id_0 = \{(P,P) \mid P \in Pr_0\}$, and $Id_1 = \{(P[x],P[x]) \mid P[x] \in Pr_1\}$:

$$S[\tau] = S \qquad S[c?] = [Id_0 \Rightarrow S] \qquad S[c!] = [Id_1 \Rightarrow S]$$

Definition 1 Bisimulation. A *bisimulation* S is a relation over Pr_0 such that whenever PSQ and $P \xrightarrow{\alpha} P'(z)$ then for some $Q'(z)$, $Q \xrightarrow{\alpha} Q'(z)$ and $P'(z)S[\alpha]Q'(z)$; and symmetrically. We denote with $\sim$ the largest bisimulation.

Up to some notational conventions $\sim$ is the bisimulation studied in [2, 14]. The relation $\sim$ is extended to process functionals by considering their equivalence on all closed instances, e.g. $P[f] \sim Q[f]$ if any $R[x]$, $[R[x]/f]P \sim [R[x]/f]Q$. Define now the function $F : Pr_0^2 \to Pr_0^2$ by $P\ F(S)\ Q$ if whenever $P \xrightarrow{\alpha} P'(z)$ then $Q \xrightarrow{\alpha} Q'(z)$ for some $Q'(z)$ and $P'(z)\ S[\alpha]\ Q'(z)$; and symmetrically. Also, let $\sim^0 = Pr_0^2$, $\sim^{\kappa+1} = F(\sim^\kappa)$, and $\sim^\lambda = \bigcap_{\kappa<\lambda} \sim^\kappa$. The relations $\sim^\kappa$ are extended to functionals following the convention for $\sim$. We obtain the following standard properties:

Proposition 2 Properties of F**.** *The set $2^{Pr_0^2}$ is a complete lattice when ordered by set inclusion. Then:*
(1) F is monotone.
(2) S is a bisimulation iff $S \subseteq F(S)$.
(3) If $\{X_i\}_{i\in I}$ is a codirected set, then $F(\bigcap_{i\in I} X_i) = \bigcap_{i\in I} F(X_i)$.
(4) The greatest bisimulation $\sim$ exists and coincides with $\sim^\omega$. □

Proposition 3 Congruence. *The relations $\sim^k$, for $k \leq \omega$, are congruences with respect to all the calculus operators. That is,*

$$\begin{array}{ll} P_i \sim^k Q_i,\ i=1,2 & \Rightarrow P_1 + P_2 \sim^k Q_1 + Q_2,\ P_1 \mid P_2 \sim^k Q_1 \mid Q_2,\ c!P_1.P_2 \sim^k c!Q_1.Q_2 \\ P \sim^k Q & \Rightarrow \nu c.P \sim^k \nu c.Q \\ P[x] \sim^k Q[x] & \Rightarrow c?x.P \sim^k c?x.Q \end{array}$$

Proof. The only difficulty arises with parallel composition. For instance, in the case $k = \omega$ one shows that $\{(\nu c_1.\cdots\nu c_n.(P \mid Q), \nu c_1.\cdots\nu c_n.(P' \mid Q)) \mid P \sim P'\}$ is a bisimulation. □

We give an alternative characterisation of the $\sim^k$ relations in terms of "sharpened" approximations, $\sim_\sharp^k$. These will be important when it comes to relating the logical and bisimulation based equivalences. These sharpened relations $\sim_\sharp^k$ are defined as follows:

$$\begin{array}{lll} P & \sim_\sharp^0\ Q & \text{always} \\ P & \sim_\sharp^{k+1}\ Q & \text{if } P \xrightarrow{\alpha} P'(z) \text{ then } Q \xrightarrow{\alpha} Q'(z) \text{ for some } Q'(z) \\ & & \text{such that } P'(z)\ \sim_\sharp^k\ Q'(z)\text{; and symmetrically} \\ P[x] & \sim_\sharp^k\ Q[x] & \text{if } P[x]\ [\sim_\sharp^k \Rightarrow \sim_\sharp^k]\ Q[x] \\ P[f] & \sim_\sharp^k\ Q[f] & \text{if } P[f]\ [[\sim_\sharp^k \Rightarrow \sim_\sharp^k] \Rightarrow \sim_\sharp^k]\ Q[f] \end{array}$$

We can now show that the sharpened approximation relations coincide with the approximations $\sim^k$. This result relies on the congruence properties of $\sim^k$.

Proposition 4. *For any $k < \omega$, $\sim^k$ coincides with $\sim^k_\sharp$.*

Proof. By induction on k and the order. □

3 Logical Characterization

Modal Formulas. Process properties are specified by the modal formulas which are generated by the following grammar, where X is a countable set. As in the case of processes, specifications also have an order. A specification of a certain order can only be predicated of a process of the same order. Conjunction and disjunction apply to formulas of the same order.

$$\phi ::= \bigwedge_{x \in X} \phi_x \,\Big|\, \bigvee_{x \in X} \phi_x \,\Big|\, \langle\alpha\rangle\phi \,\Big|\, [\alpha]\phi \,\Big|\, \phi \Rightarrow \phi$$

The truth- and falsehood constants $\top$ and $\bot$ are defined as usual: $\top = \bigwedge \emptyset$ and $\bot = \bigvee \emptyset$. These formulas are overloaded as they may have order 0, 1, and 2. We sometimes use $(\cdot)$ as a meta-connective ranging over $\{\langle\cdot\rangle, [\cdot]\}$.

Realizability. We specify when a process $P(z)$ realizes a formula ϕ, written as $\models P(z) : \phi$, by induction on the structure of ϕ. Note that a realizer of a formula $\phi \Rightarrow \psi$ is always a function, and a realizer of a modality is always a ground process.

$\models P(z) : \bigwedge_{x \in X} \phi_x$	if for all $x \in X$ $\models P(z) : \phi_x$
$\models P(z) : \bigvee_{x \in X} \phi_x$	if for some $x \in X$ $\models P(z) : \phi_x$
$\models P[z] : \phi \Rightarrow \psi$	if for all $Q(z')$, $\models Q(z') : \phi$ implies $\models [Q(z')/z]P : \psi$
$\models P : \langle\alpha\rangle\phi$	if for some $P'(z)$, $P \xrightarrow{\alpha} P'(z)$ and $\models P'(z) : \phi$
$\models P : [\alpha]\phi$	if whenever $P \xrightarrow{\alpha} P'(z)$, $\models P'(z) : \phi$

The *modal depth* $|\phi|$ of a formula ϕ is defined as follows:

$$|\bigwedge_{x \in X} \phi_x| = |\bigvee_{x \in X} \phi_x| = sup_{x \in X}|\phi_x|$$

$$|\phi \Rightarrow \psi| = |\psi| \qquad |\langle\alpha\rangle\phi| = |[\alpha]\phi| = 1 + |\phi|$$

Definition 5 Logical equivalences. We define the family of equivalence relations on process (functionals) $\sim^\kappa_L$ by $P(z) \sim^\kappa_L Q(z')$ if for all ϕ such that $|\phi| \leq \kappa$, $\models P(z) : \phi$ if and only if $\models Q(z') : \phi$. Also set: $P(z) \sim_L Q(z')$ if $P(z) \sim^\kappa_L Q(z')$, for any κ.

Proposition 6. *(1) For any κ, if $P(z) \sim^\kappa Q(z')$ then $P(z) \sim^\kappa_L Q(z')$. (2) If $P(z) \sim Q(z')$ then $P(z) \sim_L Q(z')$.*

Proof. (1) We show that if $P(z) \sim^{\kappa} Q(z')$, $|\phi| \leq \kappa$, and $\models P(z) : \phi$ then $\models Q(z') : \phi$ by induction on the structure of ϕ. The only non-standard case is when ϕ has the form $\phi_1 \Rightarrow \phi_2$. Suppose $\models P_1(z_1) : \phi_1$. Then $\models [P_1(z_1)/z]P : \phi_2$. By congruence of $\sim^{\kappa}$, $[P_1(z_1)/z]P \sim^{\kappa} [P_1(z_1)/z']Q$. By the induction hypothesis, $\models [P_1(z_1)/z']Q : \phi_2$ as desired.
(2) Immediate from (1). □

Definition 7 Characteristic formula. For any process functional $P(z)$, and ordinal $k \leq \omega$ we inductively define a formula $C^k(P(z))$:

$$C^0(P(z)) = \top$$

$$C^{k+1}(P) = \bigwedge_{P \xrightarrow{\alpha} P'(z)} (\langle\alpha\rangle.C^k(P'(z))) \wedge \bigwedge_{\alpha \in Act} ([\alpha]. \bigvee_{P \xrightarrow{\alpha} P'(z)} C^k(P'(z)))$$

$$C^{k+1}(P[z]) = \bigwedge_{R(z')} C^{k+1}(R(z')) \Rightarrow C^{k+1}([R(z')/z]P)$$

$$C^{\omega}(P(z)) = \bigwedge_{k<\omega} C^k(P(z))$$

Observe that for any $k \leq \omega$, $|C^k(P(z))| \leq k$.

Proposition 8. *For any $k < \omega$,*
(1) For all $P(z)$, $\models P(z) : C^k(P(z))$.
(2) For all $P(z), Q(z')$, $\models P(z) : C^k(Q(z'))$ iff $P(z) \sim^k Q(z')$.

Proof. One proves (1) and (2) at the same time, by induction on k and the order. We present the function case.
(1) We have to show: $\models P[x] : \bigwedge_R C^{k+1}(R) \Rightarrow C^{k+1}([R/x]P)$. Suppose that $\models R' : C^{k+1}(R)$. By the induction hypothesis and (2): $R' \sim_{k+1} R$. By congruence: $[R'/x]P \sim_{k+1} [R/x]P$. By ind. hyp. and (2): $\models [R'/x]P : C^{k+1}([R/x]P)$.
(2) Suppose $\models P[x] : C^{k+1}(Q[x])$. That is, $\models P[x] : C^{k+1}(R) \Rightarrow C^{k+1}([R/x]Q)$, any R. By ind. hyp and (1): $\models R : C^{k+1}(R)$. Hence: $\models [R/x]P : C^{k+1}([R/x]Q)$. That is: $[R/x]P \sim_{k+1} [R/x]Q$. Vice versa, suppose $P[x] \sim_{k+1} Q[x]$. Given any R, let $\models R' : C^{k+1}(R)$. Then $R' \sim_{k+1} R$, so $[R'/x]P \sim_{k+1} [R/x]P \sim_{k+1} [R/x]Q$ as desired. □

Theorem 9 Logical characterization. *For any processes P, Q,*

$$P \sim Q \;\; \textit{iff} \; P \sim_L Q.$$

Proof. Follows immediately from previous results. □

A Characterization of Chocs Bisimulation. There is an alternative and natural definition of bisimulation, which resembles the definition of sharpened approximation. Given a relation S over Pr_0^2 and an action α we define the relations $S\{\alpha\}$ as follows:

$$S\{\tau\} = S \qquad S\{c?\} = [S \Rightarrow S] \qquad S\{c!\} = [[S \Rightarrow S] \Rightarrow S]$$

Definition 10 Modified bisimulation. A modified bisimulation S is a relation over Pr_0 such that whenever PSQ and $P \xrightarrow{\alpha} P'(z)$ then for some $Q'(z)$, $Q \xrightarrow{\alpha} Q'(z)$ and $P'(z)S\{\alpha\}Q'(z)$; and symmetrically.

Proposition 11. *Among the modified bisimulations S such that $Id_0 \subseteq S$ and $[S \Rightarrow S] \subseteq Id_1$ there is a largest one and it coincides with the largest bisimulation.*

4 Towards a Proof System

As a second contribution we present a *sound* proof system to prove properties of processes stated in the *finitary fragment* of the Hennessy-Milner logic previously introduced. The following results are of a preliminary nature as they are obtained under the following strong assumptions: (1) We drop *restriction.* (2) We suppose that the calculus has a *finite number of channels.*

The restriction to finite label alphabets has two important corollaries. First, the rule PAR-BOX-τ for introducing the $[\tau]$-operator for parallel compositions becomes finitary. This condition could be lifted if, for instance, a channel quantifier was introduced into the specification language. Indeed this appears to be a natural extension. Note for example that the property that a process can perform no actions could be stated as: $\forall c.([c!]\bot \wedge [\tau]\bot \wedge [c?]\bot)$. Second, the k-th characteristic formula of any process becomes finite, for $k < \omega$. This is quite useful in arguing about the completeness of the system. We regard (1) as the main limitation of our system as in most applications one may assume a finite number of global channels.

4.1 Syntactic Conventions

Judgments. A context Γ is a set $z_1 : \psi_1, ..., z_n : \psi_n$ where all z_i are pairwise distinct. The basic judgments are sequents of the following shape: $\Gamma \vdash P : \psi$. The process P and the context Γ might contain variables of order $0, 1, 2$. There can be at most one variable which is free in P and does not occur in Γ. Following our conventions this variable should be intended as λ-abstracted (note that in our specific case this variable can be of order $0, 1$). The grammars of processes P_0, P_1, P_2 of orders $0, 1, 2$, respectively, in a context Γ can be given as follows:

$$P_0 ::= 0 \,\big|\, x \,\big|\, P_0 + P_0 \,\big|\, c!P_0.P_0 \,\big|\, c?x.P_0 \,\big|\, fP_0 \,\big|\, FP_1 \,\big|\, P_0 \mid P_0$$
$$P_1 ::= P_0[x] \qquad P_2 ::= P_0[f]$$

In the following P will denote a generic process and ψ a generic formula.

Eta-expansions. By convention we eta-expand functional variables so that: $f = fx[x], F = Ff[f]$. This allows to fit functional variables in the grammar for P_1 and P_2. In the following we will write $z_1 \mid P_0$. If $z_1 \equiv f_1$ then $z_1 \mid P_0 \equiv (f_1 x \mid P_0)[x]$, and, similarly, if $z_1 \equiv F_1$ then $z_1 \mid P_0 \equiv (F_1 f \mid P_0)[f]$ (x, f fresh variables).

Interpretation. We write $z_1 : \psi_1, ..., z_n : \psi_n \models P : \psi$ if for all closed P_i such that $\models P_i : \psi_i$ $(i = 1, ..., n)$ we have $\models [P_1/z_1, ..., P_n/z_n]P : \psi$.

4.2 Proof System

We divide the rules of the proof system (fig. 1) into *three groups*: general rules for the manipulation of the sequents, sequent calculus rules which allow for the (right and left) introduction of logical operators, and, finally, rules which exploit the process structure. Note that we have omitted the rules symmetric to AND-L, OR-R, SUM-DIA, PAR-DIA, and PAR-DIA-τ. Really, $\wedge, \vee, +$, and $\mid$ should be understood as commutative operators.

Most rules should be self-explanatory. The essential idea is that in general the holding of $\Gamma \vdash P : \phi$ depends on the structure of both P and ϕ. In all cases, but for the modal operators, P can be dealt with uniformly — these are the logical rules, and they can be seen as coming straight from proof theory. For the modal operators, however, the structure of P is essential, and its transition behaviour is exposed by the operational semantics from which the rules for the modal operators are derived in a quite systematic fashion. In formulating this last set of rules we follow to some extent previous work by Colin Stirling [15] on proof systems for CCS. The rules for parallel composition are, however, somewhat different. The rules reflect very closely the operational semantics. The most involved rules are those for parallel composition. To prove a property ϕ of a parallel composition, say, $P \mid Q$, one needs in general to: (1) *guess* properties of the parallel constituents P and Q, (2) show that they hold, and (3) show that the holding of these properties for the constituents entails the holding of ϕ for their parallel composition. We regard this as quite natural and reflecting closely the *compositional* nature of the proof system.

Example. In fig. 2 we give an example proof of the judgment $\vdash a!(b!0).b?y.c!0 \mid a?x.x : \langle\tau\rangle\langle\tau\rangle\langle c!\rangle\top$ where we have adopted the abbreviations $\phi_0 = \phi_1 \Rightarrow \phi_1$, $\phi_1 = \langle b!\rangle(\phi_2 \Rightarrow \langle c!\rangle\top)$, and $\phi_2 = \top \Rightarrow \langle c!\rangle\top$. Typically, proofs are constructed bottom up. It is useful to consider successive refinements of the formulas involved in the PAR-DIA-τ rule in fig. 2. In practice one introduces formula variables which are incrementally resolved as the proof goes on. For instance the instantiations of ϕ_0, ϕ_1 and ϕ_2 in fig. 2 have been arrived at in this way.

5 Soundness and (Infinitary) Completeness

In this section we *extend* the system (fig. 3) by an *infinitary ω-rule* which reduces the provability of open terms to the provability of their closed instances, and by

Sequent Structure Rules.

$$\textsc{Hyp}\ \frac{}{\Gamma, z : \psi \vdash z : \psi} \qquad \textsc{Cut}\ \frac{\Gamma \vdash P' : \psi' \qquad \Gamma, z : \psi' \vdash P : \psi}{\Gamma \vdash [P'/z]P : \psi}$$

Logical Rules.

$$\textsc{Bot-L}\ \frac{}{\Gamma, z : \bot \vdash P : \psi} \qquad \textsc{Top-R}\ \frac{}{\Gamma \vdash P : \top}$$

$$\textsc{And-L}\ \frac{\Gamma, z : \psi_1 \vdash P : \psi}{\Gamma, z : \psi_1 \wedge \psi_2 \vdash P : \psi} \qquad \textsc{And-R}\ \frac{\Gamma \vdash P : \psi_1 \qquad \Gamma \vdash P : \psi_2}{\Gamma \vdash P : \psi_1 \wedge \psi_2}$$

$$\textsc{Or-L}\ \frac{\Gamma, z : \psi_1 \vdash P : \psi \qquad \Gamma, z : \psi_2 \vdash P : \psi}{\Gamma, z : \psi_1 \vee \psi_2 \vdash P : \psi} \qquad \textsc{Or-R}\ \frac{\Gamma \vdash P : \psi_1}{\Gamma \vdash P : \psi_1 \vee \psi_2}$$

$$\Rightarrow\text{-L}\ \frac{\Gamma \vdash P' : \psi_1 \qquad \Gamma, x : \psi_2 \vdash P : \psi}{\Gamma, z : \psi_1 \Rightarrow \psi_2 \vdash [zP'/x]P : \psi} \qquad \Rightarrow\text{-R}\ \frac{\Gamma, z : \psi' \vdash P : \psi}{\Gamma \vdash P[z] : \psi' \Rightarrow \psi}$$

Process Structure Rules.

$$\textsc{Out-1}\ \frac{\Gamma \vdash (fP' \mid P)[f] : \psi}{\Gamma \vdash c!P'.P : (c!)\psi} \qquad \textsc{Out-2}\ \frac{\cdot}{\Gamma \vdash c!P'.P : [\alpha]\psi}\ (c! \neq \alpha)$$

$$\textsc{In-1}\ \frac{\Gamma \vdash P[x] : \psi}{\Gamma \vdash c?x.P : (c?)\psi} \qquad \textsc{In-2}\ \frac{\cdot}{\Gamma \vdash c?x.P : [\alpha]\psi}\ (c? \neq \alpha)$$

$$\textsc{Sum-dia}\ \frac{\Gamma \vdash P : \langle\alpha\rangle\psi}{\Gamma \vdash P + P' : \langle\alpha\rangle\psi} \qquad \textsc{Sum-box}\ \frac{\Gamma \vdash P : [\alpha]\psi \qquad \Gamma \vdash P' : [\alpha]\psi}{\Gamma \vdash P + P' : [\alpha]\psi}$$

$$\textsc{Nil}\ \frac{\cdot}{\Gamma \vdash 0 : [\alpha]\psi} \qquad \textsc{Par-dia}\ \frac{\Gamma \vdash P_1 : \langle\alpha\rangle\psi_1 \qquad \Gamma, z_1 : \psi_1 \vdash z_1 \mid P_2 : \psi}{\Gamma \vdash P_1 \mid P_2 : \langle\alpha\rangle\psi}$$

$$\textsc{Par-dia-}\tau\ \frac{\Gamma \vdash P_1 : \langle d!\rangle\psi_1 \qquad \Gamma \vdash P_2 : \langle d?\rangle\psi_2 \qquad z_1 : \psi_1, z_2 : \psi_2 \vdash z_1 z_2 : \psi}{\Gamma \vdash P_1 \mid P_2 : \langle\tau\rangle\psi}$$

$$\textsc{Par-box}\ \frac{\begin{array}{cc} \Gamma \vdash P_1 : [\alpha]\psi_1 & \Gamma \vdash P_2 : [\alpha]\psi_2 \\ \Gamma, z_1 : \psi_1 \vdash z_1 \mid P_2 : \psi & \Gamma, z_2 : \psi_2 \vdash P_1 \mid z_2 : \psi \end{array}}{\Gamma \vdash P_1 \mid P_2 : [\alpha]\psi}\ (\alpha \neq \tau)$$

$$\textsc{Par-box-}\tau\ \frac{\begin{array}{cc} \Gamma \vdash P_1 : [\tau]\psi_1 & \Gamma \vdash P_2 : [\tau]\psi_2 \\ \Gamma, x_1 : \psi_1 \vdash x_1 \mid P_2 : \psi & \Gamma, x_2 : \psi_2 \vdash P_1 \mid x_2 : \psi \\ \Gamma \vdash P_1 : [d?]\psi_{1,d?} \wedge [d!]\psi_{1,d!}\ (\text{all } d) & \Gamma \vdash P_2 : [d?]\psi_{2,d?} \wedge [d!]\psi_{2,d!}\ (\text{all } d) \\ z_1 : \psi_{1,d?}, z_2 : \psi_{2,d!} \vdash z_2 z_1 : \psi\ (\text{all } d) & z_1 : \psi_{1,d!}, z_2 : \psi_{2,d?} \vdash z_1 z_2 : \psi\ (\text{all } d) \end{array}}{\Gamma \vdash P_1 \mid P_2 : [\tau]\psi}$$

Fig. 1. Basic Proof System

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash 0:\top \quad x:\langle c!\rangle\top \vdash x:\langle c!\rangle\top}{g:\phi_2 \vdash g0:\langle c!\rangle\top}}{\vdash g0[g]:\phi_2 \Rightarrow \langle c!\rangle\top}}{\vdash b!0:\phi_1} \quad x:\phi_1\vdash x:\phi_1}{f:\phi_0\vdash f(b!0):\phi_1} \quad \dfrac{\dfrac{\dfrac{\vdash g0[g]:\top}{y:\top\vdash c!0:\langle c!\rangle\top}}{\vdash c!0[y]:\phi_2}}{\vdash b?y.c!0:\langle b?\rangle\phi_2}}{f:\phi_0\vdash f(b!0)\mid b?y.c!0:\langle\tau\rangle\langle c!\rangle\top}}{\vdash a!(b!0).b?y.c!0:\langle a!\rangle(\phi_0\Rightarrow\langle\tau\rangle\langle c!\rangle\top)} \quad \dfrac{\dfrac{\vdash x:\phi_1\vdash x:\phi_1}{\vdash x[x]:\phi_0}}{\vdash a?x.x:\langle a?\rangle\phi_0}}{\vdash a!(b!0).b?y.c!0 \mid a?x.x:\langle\tau\rangle\langle\tau\rangle\langle c!\rangle\top}$$

Fig. 2. Proof example

a rule stating the monotonicity of the modal operators. [3] The OMEGA rule is needed to establish a completeness result for the full specification language whereas the MON rule is needed to prove the completeness of the finitary system for a special class of so-called *well-described* specifications (to be described in the next section). First, however, we prove the soundness of the extended system.

$$\text{OMEGA}\ \frac{\text{For all } P' \text{ such that } \models P':\psi',\ \Gamma\vdash[P'/z]P:\psi}{\Gamma, z:\psi'\vdash P:\psi}$$

$$\text{MON}\ \frac{\Gamma, z:\phi\vdash z:\psi}{\Gamma, x:(\alpha)\phi\vdash x:(\alpha)\psi}$$

Fig. 3. Additional Rules for Completeness Results.

Proposition 12 Soundness. *If $\Gamma\vdash P:\psi$ then $\Gamma\models P:\psi$.*

Proof. Proofs are well-founded, countably branching trees. To every proof one can associate a (transfinite) ordinal which measures its depth. Proceed by transfinite induction on the proof depth. □

Proposition 13. *Suppose that there are a finite number of channels. Then, for any process P and number $k<\omega$: (1) $C^k(P)$ is a finite formula, (2) $\{C^k(P)\mid P$ process$\}$ is a finite set (up to identification of ψ with $\psi\wedge\psi$).*

Proof. Prove 1 and 2 together by induction on k and P order. □

Theorem 14 Completeness for closed processes. *If $\models P:\psi$ then $\vdash P:\psi$.*

[3] Note that in fact HYP is derivable using MON. This can be seen using a little structural induction.

Proof. Induction on the lexicographic order $(|\psi|, order(\psi), struct(P), struct(\psi))$. One proceeds by case analysis on the structure of P and ψ.

1. $\psi ::= \top \mid \bot$: Direct.
2. $\psi ::= \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2$: $struct(\psi)$ decreases.
3. $\psi ::= \psi_1 \Rightarrow \psi_2$: $order(\psi)$ decreases, use ω-rule.
4. $\psi ::= (\alpha)\psi$: We analyse the structure of the process P (which has order 0).
   (a) $P ::= 0$: Direct.
   (b) $P ::= c!P.P \mid c?x.P$: $|\psi|$ decreases.
   (c) $P ::= P + P$: $struct(P)$ decreases.
   (d) $P ::= P \mid P$: There are two subcases.
      i. $\psi ::= \langle\alpha\rangle\psi$: We give this case in some detail.
      • Suppose $\models P_1 \mid P_2 : \langle\alpha\rangle\psi$ because $P_1 \xrightarrow{\alpha} P_1'$ and $\models P_1' \mid P_2 : \psi$. Let $k = |\psi|$ and $\psi_1 = C^k(P_1')$. We know: $\models P_1' : \psi_1$. Hence: $\models P_1 : \langle\alpha\rangle\psi_1$. We can conclude $\vdash P_1 : \langle\alpha\rangle\psi_1$, by ind. hyp. on P.
      Next we show: $z_1 : \psi_1 \models z_1 \mid P_2 : \psi$. Suppose $\models P_1'' : \psi_1$, then $P_1'' \sim_k P_1'$, which implies $P_1'' \mid P_2 \sim_k P_1' \mid P_2$. Conclude: $\models P_1'' \mid P_2 : \psi$. By induction on $|\psi|$ we have: $\vdash P_1'' \mid P_2 : \psi$. By the ω-rule and PAR-DIA we prove: $\vdash P_1 \mid P_2 : \langle\alpha\rangle\psi$.
      • Otherwise suppose $\alpha \equiv \tau$ and $\models P_1 \mid P_2 : \langle\tau\rangle\psi$ because $P_1 \xrightarrow{d!} P_1'[f]$, $P_2 \xrightarrow{d?} P_2'[x]$, and $\models [P_2'[x]/f]P_1' : \psi$. Let $k = |\psi|$, $\psi_1 = C^k(P_1'[f])$, and $\psi_2 = C^k(P_2'[x])$, Clearly: $\models P_1 : \langle d!\rangle\psi_1$ and $\models P_2 : \langle d?\rangle\psi_2$. Conclude: $\vdash P_1 : \langle d!\rangle\psi_1$ and $\vdash P_2 : \langle d?\rangle\psi_2$, by ind. hyp. on P.
      It remains to show $z_1 : \psi_1, z_2 : \psi_2 \models z_1 z_2 : \psi$. Apply again the logical characterization of the $\sim_k$ relation. Then apply twice the ω rule to get: $z_1 : \psi_1, z_2 : \psi_2 \vdash z_1 z_2 : \psi$. Conclude $\vdash P_1 \mid P_2 : \langle\tau\rangle\psi$, by PAR-DIA-$\tau$.
      ii. $\psi ::= [\alpha]\psi$: This behaves as the previous case w.r.t. the induction hypothesis. □

Remark. Note that this proof never uses the left introduction rules (as hypotheses on the left of the sequent are eliminated by means of the ω-rule). Of course a *finitary* system makes essential use of the left introduction rules, as in fig. 2.

Corollary 15 Completeness. *If $\Gamma \models P : \psi$ then $\Gamma \vdash P : \psi$.* □

6 Finitary Completeness for Well-described Specifications

In this section we seek a finer evaluation of the power and weakness of the finitary system, i.e. the basic proof system in fig. 1 plus the MON rule. The guiding idea is that the system is complete provided specifications are presented in an "explicit" way. We intend to capture this idea by the concepts of k-determinedness, and well-describedness.

Definition 16. A formula ψ is *k-determined* ($k < \omega$) if there are $P_1(z), ..., P_n(z)$, $n \geq 1$, such that $\psi \equiv C^k(P_1(z)) \vee ... \vee C^k(P_n(z))$.

Definition 17. The class of *well-described* formulas is determined as follows:

$\top, \bot$ are well-described.
$\psi_1 \vee \psi_2, \psi_1 \wedge \psi_2$ are well-described if ψ_1, ψ_2 are.
$(\alpha)\psi$ is well-described if ψ is well-described.
$\psi_1 \Rightarrow \psi_2$ is well-described if ψ_2 is well-described and ψ_1 is k-determined, $|\psi_2| \leq k$.

Remarks.

1. The intention is to show that the proof system is complete on all well-formed judgments $z_1 : \psi_1, ..., z_n : \psi_n \vdash P : \psi$ such that $\psi_1 \Rightarrow ... \Rightarrow \psi_n \Rightarrow \psi$ is well-described. By convention we will call these judgments well-described.
2. Every k-determined formula is well-described (easy induction). However, there are well-described formulas which are not k-determined, for instance: $\langle \tau \rangle \top$.
3. The notion of k-determined formula (and therefore also the notion of well-described formula) is not invariant under interpretation equivalent formulas. For instance, $\top$ and $\bigvee_p C^1(p)$ are equivalent formulas but in general the former is not 1-determined (note that the second formula is finite modulo identification of $\psi \vee \psi$ with ψ).
4. Note that a k-determined formula is always realizable.

On well-described judgments the validity of a disjunction of formulas can always be reduced to the validity of one of the formulas. This property plays an important role in the following completeness proof.

Proposition 18. *If* $\Gamma \models P : \psi_1 \vee \psi_2$ *then* $\Gamma \models P : \psi_1$ *or* $\Gamma \models P : \psi_2$.

Proof. Suppose $|\psi_1 \vee \psi_2| = k$. W.l.o.g we may assume (the disjunction on the left can be eliminated): $z_1 : C^k(P_1(z)), ..., z_n : C^k(P_n(z)) \models P : \psi_1 \vee \psi_2$.
This implies: $\models [\mathbf{P_i(z)/z_i}]P : \psi_j$ for $j = 1$ or $j = 2$. Say $j = 1$, then:
$z_1 : C^k(P_1(z)), ..., z_n : C^k(P_n(z)) \models P : \psi_1$ as if $\models Q_i(z) : C^k(P_i(z)), i = 1, ..., n$ then $[\mathbf{Q_i(z)/z_i}]P \sim^k [\mathbf{P_i(z)/z_i}]P$ and therefore $\models [\mathbf{Q_i(z)/z_i}]P : \psi_1$. □

Theorem 19 Completeness on well-described judgments. *Suppose* $\Gamma \models P : \psi$ *and the judgment is well-described. Then* $\Gamma \vdash P : \psi$ *is provable without the* ω*-rule.*

Proof. Suppose $z_1 : \psi_1, ..., z_n : \psi_n \models P : \psi$. As in Theorem 14 proceed by induction on the lexicographic order: $(|\psi|, order(\psi), struct(P), struct(\psi))$ and by case analysis on the structure of P and ψ. We only discuss cases which differ in some important way from the corresponding case in the proof of Theorem 14.

1. $\psi ::= \psi_1 \vee \psi_2$: $struct(\psi)$ decreases. By the disjunctive property follows $\Gamma \models P : \psi_i$, say for $i = 1$. By ind. hyp. $\Gamma \vdash P : \psi_1$. Conclude by OR-R.

2. $\psi ::= \psi \Rightarrow \psi'$: $\Gamma \models P[z] : \psi \Rightarrow \psi'$ iff $\Gamma, z : \psi \models P : \psi'$, and $order(\psi)$ decreases.
3. $\psi ::= (\alpha)\psi$: We analyse the structure of the process P (which has order 0).
   (a) $P ::= x$: This case and the next one do not arise in the completeness proof for the infinitary system. W.l.o.g suppose $x : C^k(q) \models x : (\alpha)\psi$. Then $\models q : (\alpha)\psi$. Distinguish two cases:
      i. $(\alpha) \equiv \langle\alpha\rangle$. Then $q \xrightarrow{\alpha} q'$ and $\models q' : \psi$. Hence $C^k(q) \equiv \langle\alpha\rangle C^{k-1}(q') \wedge \psi'$, for some ψ'. It follows: $x : C^{k-1}(q') \models x : \psi$. By ind. hyp. $x : C^{k-1}(q') \vdash x : \psi$. Conclude by MON plus AND-R .
      ii. $(\alpha) \equiv [\alpha]$. If q can make no α transition then $C^k(q) \equiv [\alpha]\bot \wedge \psi'$, for some ψ'. Conclude from $z : \bot \vdash z : \psi$. Otherwise, suppose $q \xrightarrow{\alpha} q'$ then $x : C^{k-1}(q') \models x : \psi$. Conclude as in the previous case.

   (b) $P ::= zP$: W.l.o.g. suppose $\Gamma, z : C^k(q[z']) \models zP : (\alpha)\psi$, where $\Gamma \equiv z_1 : C^k(q_1(z')), ..., z_n : C^k(q_n(z'))$. Note that z does not occur in P. Let $r \equiv [\mathbf{q}/\mathbf{z}]P$. Then $\models q[r] : (\alpha)\psi$. Hence: $x : C^k(q[r]) \models x : (\alpha)\psi$ and $\Gamma \models P : C^k(r)$. By ind. hyp. ($struct(P)$ decreases) $x : C^k(q[r]) \vdash x : (\alpha)\psi$ and $\Gamma \vdash P : C^k(r)$. By $\Rightarrow$-L $\Gamma, z : C^k(r) \Rightarrow C^k(q[r]) \vdash zP : (\alpha)\psi$. By AND-L $\Gamma, z : C^k(q[z']) \vdash zP : (\alpha)\psi$. □

7 Research Directions

The issue of finitary completeness, and the attempt to expand the finitary completeness result to richer types of sequent is worth further investigation. We discuss this through a couple of examples.

Example 1. Consider the following valid judgment:

$$x : \top \vdash x : \langle\tau\rangle\top \vee [\tau]\bot$$

It is easy to see that this judgment cannot be proved in the finitary proof system as it stands. One solution could be the inclusion of a rule of consequence allowing the inference of $\Gamma, x : \phi \vdash x : \psi$ whenever $\phi \supset \psi$ is a a theorem of the modal logic K (c.f. [16]). Note that K-theoremhood is decidable. This would allow inference of a number of distribution-like properties such as $\wedge$-$\vee$ distribution and distribution of $\vee$ through $\langle\alpha\rangle$ which are not presently derivable. However we have currently no completeness result for this inference system.

Example 2. Next let us consider a more subtle valid judgment:

$$f : (1_\tau \Rightarrow 2_\tau) \wedge (2_\tau \Rightarrow 1_\tau) \vdash f : \bot$$

where $1_\tau \equiv C(\tau)$ and $2_\tau \equiv C(\tau.\tau)$ (C is the characteristic formula). Proving this judgment amounts to realise that there is no process function $p[x]$ which can separate the processes τ and $\tau.\tau$ without trying to execute them. A proof of this fact should rely on the structure of closed processes. For instance, one may consider adding an induction principle which analyses the structure of process (functionals).

Finally, yet another interesting problem which remains to be settled is that of developing a proof system which can handle restriction. In order to appreciate the difficulties, one may try to develop rules to prove the following valid fact, where Nil is a formula stating that a process can do no action:

$$\nu a.(b!(a?x.0).a?x.0) : \langle a!\rangle(Nil \Rightarrow Nil) \Rightarrow Nil.$$

Acknowledgments. We are indebted to L. Leth, S. Prasad, and B. Thomsen for several discussions on the topics presented here.

References

1. S. Abramsky. A domain equation for bisimulation. *Information and Computation*, 92:161–218, 1991.
2. R. Amadio. On the reduction of chocs bisimulation to π-calculus bisimulation. In *Proc. CONCUR 93, Hildesheim*, pages 112–126. SLNCS 715, 1993. Also appeared as Research Report Inria-Lorraine 1786, October 1992.
3. G. Boudol. Towards a lambda calculus for concurrent and communicating systems. *SLNCS*, 351, 1989. In Proc. TAPSOFT.
4. M. Dam. Model checking mobile processes. In *Proc. CONCUR'93,* Lecture Notes in Computer Science, 715:22–36, 1993. Full version in SICS report RR94:1, 1994.
5. A. Giacalone, P. Mishra, and S. Prasad. Facile: A symmetric integration of concurrent and functional programming. *International Journal of Parallel Programming*, 18(2):121–160, 1989.
6. M. Hennessy. A denotational model for higher-order processes. In *Proc. IEEE-LICS*, 1993.
7. M. Hennessy and X. Liu. A modal logic for message passing processes. Dept. of Computer Science, University of Sussex, Report 3/93, 1993.
8. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, **32**:137–162, 1985.
9. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
10. R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *TCS*, 114:149–171, 1993.
11. F. Nielsen. The typed lambda calculus with first class processes. *Springer Lecture Notes in Computer Science*, 366, 1989. In Proc. PARLE.
12. A. Pitts and I. Stark. What's new? In *Proc. Mathematical Foundations of Computer Science, Gdańsk, Poland.* SLNCS 711, 1993.
13. J. Reppy. Cml: A higher-order concurrent language. In *Proc. ACM-SIGPLAN 91, Conf. on Prog. Lang. Design and Impl.*, 1991.
14. D. Sangiorgi. *Expressing mobility in process algebras: first-order and higher order paradigms.* PhD thesis, University of Edinburgh, September 1992.
15. C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–347, 1987.
16. C. Stirling. Modal and temporal logics. In *Handbook of Logic in Computer Science* Vol. 2, Oxford University Press, 1992.
17. B. Thomsen. *A calculus of higher order communicating systems.* PhD thesis, Imperial College, London, 1990.
18. B. Thomsen. Plain chocs. *Acta Informatica*, 30:1–59, 1993. Also appeared as TR 89/4, Imperial College, London.