SICS research report R96:03
ISRN: SICS-R–96/3–SE
ISSN: 0283-3638

Toward a modal theory of types for the π-calculus

Roberto M. Amadio
CNRS, BP 145
Sophia-Antipolis, F-06903, France
e-mail: amadio@cma.cma.fr

Mads Dam[1]
SICS
Box 1263, Kista, Sweden
e-mail: mfd@sics.se

Keywords: Process algebra, temporal logic, mobile processes, compositional verification, model checking

[1]Work partially supported by Esprit BRA project 8130 LOMAPS

Abstract

We study the problem of specifying and verifying properties of π-calculus processes while relying on a bisimulation semantics. As our property specification language we use a version of the modal μ-calculus adapted to the π-calculus. We show that the logical language is sufficiently *expressive* to characterize by means of a finite formula a process up to any approximation of the bisimulation relation. We consider the problem of *checking* that a process of the π-calculus satisfies a specification expressed in this modal μ-calculus. We develop an algorithm which is sound in general, and complete for processes having a *finite reachability* property. Finally, we present a proof system which can be applied to prove non-recursive properties of arbitrary processes. We show that the system is complete on the non-recursive fragment of the logical language.

1 Introduction

The π-calculus was introduced by Milner, Parrow, and Walker [MPW92] as a language for describing concurrent systems with features such as mobility, parameter dependency, and dynamic reconfigurability that were outside the realm of CCS and related formalisms. Essentially, the π-calculus augments CCS by adding to it facilities for generating new names of communication channels, and for passing channel names between concurrent processes. The π-calculus has been used successfully in a range of applications, from studies of reduction strategies in the λ-calculus [Mil92, San92, ALT95] and true concurrency semantics [AP94, San94] to modellings of telecommunication protocols [Ora94] and object-oriented programming languages [Wal94]. In this paper we study the problem of specifying and verifying properties of π-calculus processes. This is a rather delicate issue for two reasons:

1. While Hennessy-Milner logic [HM85] and the modal μ-calculus [Koz83] have by now emerged as standard basic devices for specifying properties of labelled transition systems no corresponding tools have as yet emerged to account for value passing or channel generation.

2. With some exceptions (e.g. the hand-over protocol study of Orava and Parrow [OP92]) most interesting applications of the π-calculus give rise to dynamic process networks which are not easily analysed using the verification technology of the day.

As our property specification language we use a version of the modal μ-calculus adapted to the π-calculus. Roughly, the language is the Hennessy-Milner logic considered in [MPW93] extended with:

(i) A generalized box operator stating that a co-finite collection of actions cannot happen.

(ii) Least and greatest fixed point operators.

Both extensions allow properties to be expressed that cannot be described by a *finite* formula of the logic studied in [MPW93]. The generalized box operator can be used to characterize a process up to any approximation of the bisimulation relation (theorem 1). This fact is quite useful in the construction of a *proof system* (theorem 4). Fixed point operators can be used to characterize processes having a finite reachability property (FR-processes for short) up to bisimulation (theorem 3) and to express *weak* modalities.

We consider the problem of *checking* that a process of the π-calculus satisfies a specification expressed in this modal μ-calculus. Following work by Winskel [Win89] on CCS (see also [SW91]) and Dam [Dam93] on the π-calculus we develop an algorithm which is sound in general, and complete for FR-processes. The algorithm is formalized by inference rules and relies on a suitable generalization of

the well-known technique of *tagging* fixed points. The key to the soundness proof is a generalization to functions of Kozen's reduction lemma [Koz83]. Completeness is proven by exhibiting a bound on the size of the proofs of a judgment. This bound is obtained as a corollary of the termination of a simple rewriting system. The strong normalization proof of this rewriting system highlights the core of the combinatorial problem.

The termination conditions embedded in the model checker are inherently limited to models whose global state space can be bounded. For dynamic process networks which do not in general possess the FR-property the model checker can only prove limited properties that depend only on a finite part of an otherwise infinite state space (as the compositional model checker of [ASW94]). To overcome this problem we embark on a study of *compositional proof systems* that are not a priori restricted to FR processes. Following work presented in [AD95, Dam95, Sti87] we define a proof system operating on judgments $\Gamma \vdash^s p : \phi$ where Γ is a sequence of the form $x_1 : \phi_1, ..., x_n : \phi_n$ and s is a finite set of channel names. Such a judgment states properties ϕ of the process p relative to properties ϕ_i of its components, x_i, under a suitable restriction of the names in s. We present a system which is sound in general and complete on recursion free formulas. The proof system can also be regarded as a *type system* for the π-calculus in the sense that it provides approximate descriptions of the dynamic behaviour of a process. Extensions of the proof system to recursive formulas is left for future work.

Our priorities in designing the system have been to keep both the process and property specification languages as simple as possible by, for instance, avoiding polyadic channels and operators such as matching, mismatching, and quantifiers (which were considered in [Dam93]), and by concentrating on one sort of bisimulation equivalence: the one based on early instantiation ([MPW93] considers various subcases). In spite of this, the proof system is quite unwieldy, containing in the order of 40 rules. Though the rules can be logically organized into various subgroups there is a lot of cases to be considered. Roughly one has to examine for each of the two modalities six process combinators, and for each of these 5 types of transition labels. Future work will have to give evidence for the practical relevance of the proof system. In fact the proof system should be viewed as an attempt to pin down a sort of assembly language for reasoning about mobile processes. Real applications will require more complex and generic rules and partial automation.

The paper is organized as follows: section 2 recalls basic π-calculus notation, section 3 introduces a variant of the μ-calculus adapted to the specification of π-processes, section 4 describes the model checker, and section 5 presents the proof system. Finally research directions are mentioned in section 6.

2 Preliminaries on the π-calculus

We introduce some notation and some standard results on the π-calculus. Let Ch be a countable collection of channel names, say $a, b, \ldots$ All free names should be

regarded as constants. Processes are specified by the following grammar:

$$p ::= 0 \mid a(b).p \mid \overline{a}b.p \mid \nu a.p \mid p + p \mid p \mid p \mid (recA(\vec{a}).p)(\vec{b}) \mid A(\vec{a}) \mid x$$

Remarks:

- In $a(b).p$ and $\nu b.p$ we assume that the name b is bound in p. Processes are considered up to renaming of bound names. The resulting congruence is denoted with $\equiv$. We denote with $fn(p)$ and $bn(p)$ the free and bound names occurring in p, respectively. All free names should be regarded as constants.

- Agent identifiers are ranged over by $A, B, \ldots$ In a well formed process all agent identifiers are bound by rec. In the term $(recA(\vec{a}).p)$ we suppose that all occurrences of agent identifiers in p are guarded (i.e. preceeded by a prefix), and $fn(p) \subseteq \{\vec{a}\}$.

- $x, y, \ldots$ denote process variables. We denote with $FV(p)$ the process variables occurring free in the process p. Up to section 5, we assume that processes do not contain process variables.

Actions are specified by the following grammar: [1] $\alpha ::= \tau \mid ab \mid \overline{a}b \mid a(b) \mid \overline{a}(b)$. Conventionally we set $n(\alpha) = fn(\alpha) \cup bn(\alpha)$ where:

$$\begin{array}{llll} fn(\tau) = \emptyset & fn(\overline{a}(b)) = fn(a(b)) = \{a\} & fn(\overline{a}b) = fn(ab) = \{a, b\} \\ bn(\tau) = \emptyset & bn(\overline{a}(b)) = bn(a(b)) = \{b\} & bn(\overline{a}b) = bn(ab) = \emptyset \end{array}$$

The labelled transition system (lts) follows an early instantiation schema and it is described in figure 1, where we omit the symmetric version of the rules $(sync)$, $(sync_{ex})$, $(comp)$, and (sum).

Definition 1 (bisimulation) Let Pr be the collection of (closed) processes. We define an operator $\mathcal{F} : \mathcal{P}(Pr \times Pr) \to \mathcal{P}(Pr \times Pr)$ as:

$$\begin{array}{ll} p\mathcal{F}(S)q & \text{if } \forall p' \forall \alpha \, (bn(\alpha) \cap fn(q) = \emptyset \text{ and } p \xrightarrow{\alpha} p') \\ & \text{implies } \exists q' \, (q \xrightarrow{\alpha} q' \text{ and } p' S q') \qquad \text{(and symmetrically)} \end{array}$$

A relation S is a bisimulation if $S \subseteq \mathcal{F}(S)$. We define:

$$\sim = \bigcup\{S \mid S \subseteq \mathcal{F}(S)\} \qquad \sim^0 = Pr \times Pr \qquad \sim^{k+1} = \mathcal{F}(\sim^k)$$

The following are well-known properties of bisimulation for the π-calculus.

Proposition 1 *1. Let η be an injective substitution on $fn(p \mid q)$ then $p \sim q$ iff $\eta p \sim \eta q$.*

2. The operator $\mathcal{F}$ preserves filtered intersections, in particular $\sim = \bigcap_{k<\omega} \sim^k$.

[1] The action $a(b)$ represents the input of a new name b on a channel a. This action is taken as a *derived* action in standard accounts of the π-calculus. Namely one states: $p \xrightarrow{a(b)} p'$ if $p \xrightarrow{ab} p'$ and $b \notin fn(p)$. We found it more convenient to introduce $a(b)$ as a *primitive* action as it allows for a symmetric and synthetic formulation of certain proof rules (cf. section 5).

$$(in)\quad \frac{\cdot}{a(b).p \xrightarrow{a\,c} [c/b]p} \qquad (out)\quad \frac{\cdot}{\overline{a}b.p \xrightarrow{\overline{a}b} p}$$

$$(in_{ex})\quad \frac{\cdot}{a(b).p \xrightarrow{a(b)} p} \qquad (out_{ex})\quad \frac{p \xrightarrow{\overline{a}b} p' \quad a \neq b}{\nu b.p \xrightarrow{\overline{a}(b)} p'}$$

$$(sync)\quad \frac{p \xrightarrow{\overline{a}b} p' \quad q \xrightarrow{ab} q'}{p \mid q \xrightarrow{\tau} p' \mid q'} \qquad (sync_{ex})\quad \frac{p \xrightarrow{\overline{a}(b)} p' \quad q \xrightarrow{a(b)} q'}{p \mid q \xrightarrow{\tau} \nu b.(p' \mid q')}$$

$$(\nu)\quad \frac{p \xrightarrow{\alpha} p' \quad a \notin n(\alpha)}{\nu a.p \xrightarrow{\alpha} \nu a.p'} \qquad (comp)\quad \frac{p \xrightarrow{\alpha} p' \quad bn(\alpha) \cap fn(q) = \emptyset}{p \mid q \xrightarrow{\alpha} p' \mid q}$$

$$(sum)\quad \frac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{\alpha} p'} \qquad (rec)\quad \frac{[(recA(\vec{a}).p)/A, \vec{b}/\vec{a}]p \xrightarrow{\alpha} p'}{(recA(\vec{a}).p)(\vec{b}) \xrightarrow{\alpha} p'}$$

$$(cong)\quad \frac{p \equiv p' \quad p' \xrightarrow{\alpha} q' \quad q' \equiv q}{p \xrightarrow{\alpha} q}$$

Figure 1: Labelled transition system

3 An extended modal μ-calculus

We introduce a Hennessy-Milner logic for the π-calculus and study its interpretation. The logic includes least and greatest fixed points and a generalized box-operator roughly saying that a co-finite collection of actions cannot happen. First we need to fix some notation on actions.

- Let β vary over the set: $B = \{a \mid a \in Ch\} \cup \{\overline{a} \mid a \in Ch\} \cup \{\overline{a}b \mid a, b \in Ch\}$.
- Let γ vary over the co-finite subsets of B, that is $\gamma = \{\beta_1, \ldots, \beta_n\}^c = B \backslash \{\beta_1, \ldots, \beta_n\}$ where $n \geq 0$ and $\beta_i \in B$. Given an action α we define its *projection* $prj(\alpha) \in B$ as follows:

  $$prj(ab) = a \quad prj(a(b)) = a \quad prj(\overline{a}b) = \overline{a}b \quad prj(\overline{a}(b)) = \overline{a}$$

  Given a process p we define the collection $pact(p) \subseteq B$ of its projected actions as follows:

  $$pact(p) = \{prj(\alpha) \mid \alpha \neq \tau \text{ and } p \xrightarrow{\alpha} p'\}$$

  Observe that the set $pact(p)$ is always finite. We also define the subject of an input-output action as follows:

  $$subj(ab) = subj(\overline{a}b) = subj(a(b)) = subj(\overline{a}(b)) = a$$

- We assume that the set of channels Ch is linearly ordered and that the function *fst* returns the least channel in a (non-empty) set. We define:

  $$\begin{array}{ll} Fout(p) & = \{\overline{a}b \mid p \xrightarrow{\overline{a}b} p'\} \\ Fin(p) & = \{ab \mid p \xrightarrow{ab} p' \text{ and } b \in fn(p)\} \\ Bout(p) & = \{\overline{a}(b) \mid p \xrightarrow{\overline{a}(b)} p' \text{ and } b = fst(Ch \backslash fn(p))\} \\ Bin(p) & = \{a(b) \mid p \xrightarrow{a(b)} p' \text{ and } b = fst(Ch \backslash fn(p))\} \\ \tau(p) & = \{\tau \mid p \xrightarrow{\tau} p'\} \\ act(p) & = Fout(p) \cup Fin(p) \cup Bout(p) \cup Bin(p) \cup \tau(p) \end{array}$$

  Observe that the set $act(p)$ is finite.

Assume an infinite collection of formula variables $X, Y, \ldots$ each with a specifed arity $ar(X) \geq 0$. A *tag* of arity n is a set $\{\vec{a}_i \Rightarrow p_i\}_{i \in I}$ where $\vec{a}_i \in Ch^n$ and $p_i \in Pr$. Tags are a technical device which plays an important role in the following proofs. We denote tags with $t, t', \ldots$ A tag t of arity n can be seen as a function in $Ch^n \to \mathcal{P}(Pr)$ by defining $t(\vec{b}) = \{p \mid (\vec{b} \Rightarrow p) \in t\}$.

The functional space $Ch^n \to \mathcal{P}(Pr)$ inherits from $\mathcal{P}(Pr)$ the order and the set theoretic operations. In particular we have:

$$\begin{array}{lll} f \leq g & \text{iff } \forall \vec{a}\,(f(\vec{a}) \subseteq g(\vec{a})) & (f \backslash g)(\vec{a}) = f(\vec{a}) \backslash g(\vec{a}) \\ (f \cup g)(\vec{a}) & = f(\vec{a}) \cup g(\vec{a}) & (f \cap g)(\vec{a}) = f(\vec{a}) \cap g(\vec{a}) \end{array}$$

The space $Ch^n \to \mathcal{P}(Pr)$ is a complete lattice ordered by $\leq$. In particular monotone functions over this space have a least fixed point (lfp) and a greatest fixed point (gfp). The following proposition generalizes to order 1 a remark by Kozen, later exploited by Winskel in the definition of a model checker.

Proposition 2 *Let F be a monotone function over $Ch^n \to \mathcal{P}(Pr)$. Then*

$$\begin{array}{ll}(1) & \vec{b} \Rightarrow p \leq lfp(F) \quad iff \quad \vec{b} \Rightarrow p \leq F(lfp(\lambda f.Ff \backslash \{\vec{b} \Rightarrow p\})) \\ (2) & \vec{b} \Rightarrow p \leq gfp(F) \quad iff \quad \vec{b} \Rightarrow p \leq F(gfp(\lambda f.Ff \cup \{\vec{b} \Rightarrow p\})) \\ (3) & \vec{b} \Rightarrow p \not\leq lfp(\lambda f.Ff \backslash \{\vec{b} \Rightarrow p\}) \\ (4) & \vec{b} \Rightarrow p \leq gfp(\lambda f.Ff \cup \{\vec{b} \Rightarrow p\})\end{array}$$

PROOF. The interesting part of (1) is to prove the direction ($\Rightarrow$). Let F^δ, for δ ordinal, be the δ^{th} iterate of function F. Let δ_o be the least ordinal such that $\vec{b} \Rightarrow p \leq F^{\delta_o}$. Note that $\delta_o = \delta' + 1$, for some δ'. Then we show that:

$$\forall \delta < \delta_o \, F^\delta \leq lfp(\lambda f.Ff \backslash \{\vec{b} \Rightarrow p\})$$

By monotonicity it follows that: $F^{\delta_o} = F(F^{\delta'}) \leq F(lfp(\lambda f.Ff \backslash \{\vec{b} \Rightarrow p\}))$. (2) follows by a dual argument. (3-4) are proven by unfolding the fixed point. □

Now we define the formulas of our logical language:

$$\phi ::= \bot \mid \top \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle\alpha\rangle\phi \mid [\alpha]\phi \mid [\gamma]\bot \mid X(\vec{c}) \mid (\sigma X(\vec{a}), t.\phi)(\vec{b}) \qquad (1)$$

Here σ stands for either μ, the lfp, or ν, the gfp. $(\alpha)\phi$ is a generic notation for both $\langle\alpha\rangle\phi$ and $[\alpha]\phi$. In $(a(b))\phi$ and $(\overline{a}(b))\phi$ the name b is bound. Formulas are considered up to alpha renaming. A formula is *closed* if it does not contain free formula identifiers. The modal depth $|\phi|$ of a formula ϕ is an ordinal inductively defined as follows:

$$\begin{array}{ll}|\top| = |\bot| = 0 & |\phi \wedge \psi| = |\phi \vee \psi| = max\{|\phi|, |\psi|\} \\ |(\alpha)\phi| = 1 + |\phi| & |[\gamma]\bot| = 1 \\ |(\sigma X(\vec{a}).\phi)(\vec{b})| = \omega & \end{array}$$

Next we define a formula interpretation. Let $Fide$ be the collection of formula identifiers. An *environment* ρ is an element of $\Pi_{X \in Fide}(Ch^{ar(X)} \to \mathcal{P}(Pr))$, so $\rho(X) : Ch^{ar(X)} \to \mathcal{P}(Pr)$. The interpretation of a formula is given parametrically w.r.t. an environment ρ in figure 2. In this context the condition "d fresh" means that d is chosen such that d is neither in $fn(p)$ or $fn((\alpha)\phi)$ If ϕ has no free agent identifiers then the interpretation is independent from the environment, and we write $\models p : \phi$ if $p \in [\![\phi]\!]\rho$, for some ρ. To claim that the interpretation actually computes the least and greatest fixed points of the functionals M and N we have to prove that M and N are monotone. This follows from the following proposition.

$$[\![\bot]\!]\rho = \emptyset$$

$$[\![\top]\!]\rho = Pr$$

$$[\![\phi \wedge \psi]\!]\rho = [\![\phi]\!]\rho \cap [\![\psi]\!]\rho$$

$$[\![\phi \vee \psi]\!]\rho = [\![\phi]\!]\rho \cup [\![\psi]\!]\rho$$

$$[\![\langle\alpha\rangle\phi]\!]\rho = \begin{cases} \{p \mid p \xrightarrow{\alpha} p', p' \in [\![\phi]\!]\rho\} & \text{if } bn(\alpha) = \emptyset \\ \{p \mid p \xrightarrow{\alpha} p', [d/c]p' \in [\![[d/b]\phi]\!]\rho, d \text{ fresh}\} & \text{if } bn(\alpha) = \{c\} \end{cases}$$

$$[\![[\alpha]\phi]\!]\rho = \begin{cases} \{p \mid p \xrightarrow{\alpha} p' \text{ implies } p' \in [\![\phi]\!]\rho\} & \text{if } bn(\alpha) = \emptyset \\ \{p \mid p \xrightarrow{\alpha} p' \text{ implies } [d/c]p' \in [\![[d/b]\phi]\!]\rho, d \text{ fresh}\} & \\ & \text{if } bn(\alpha) = \{c\} \end{cases}$$

$$[\![[\{\beta_1, \ldots, \beta_m\}^c]\bot]\!]\rho = \{p \mid pact(p) \subseteq \{\beta_1, \ldots, \beta_m\}\}$$

$$[\![X(\vec{b})]\!]\rho = \rho(X)(\vec{b})$$

$$[\![(\mu X(\vec{a}), t.\phi)(\vec{b})]\!]\rho = (\bigcap\{f : Ch^{ar(X)} \to \mathcal{P}(Pr) \mid M(f) \leq f\})(\vec{b})$$
where $M(f)(\vec{c}) = [\![[\vec{c}/\vec{a}]\phi]\!]\rho[f/X]\backslash t(\vec{c})$

$$[\![(\nu X(\vec{a}), t.\phi)(\vec{b})]\!]\rho = (\bigcup\{f : Ch^{ar(X)} \to \mathcal{P}(Pr) \mid f \leq N(f)\})(\vec{b})$$
where $N(f)(\vec{c}) = [\![[\vec{c}/\vec{a}]\phi]\!]\rho[f/X] \cup t(\vec{c})$

Figure 2: Interpretation

Proposition 3 *For any formula ϕ, formula identifier X, and environment ρ, if $f \leq g$ in $Ch^{ar(X)} \to \mathcal{P}(Pr)$ then $[\![\phi]\!]\rho[f/X] \subseteq [\![\phi]\!]\rho[g/X]$.*

PROOF. By induction on ϕ size. Let us consider two interesting cases. If ϕ is a formula identifier the hypothesis $f \leq g$ applies. If ϕ is a least fixed point formula, say $(\mu Y(\vec{a}), t.\phi)(\vec{b})$, then it is enough to prove that $M(h)(\vec{c}) \leq M'(h)(\vec{c})$, where:

$$\begin{array}{ll} M(h)(\vec{c}) & = [\![[\vec{c}/\vec{a}]\phi]\!]\rho[f/X, h/Y]\backslash t \\ M'(h)(\vec{c}) & = [\![[\vec{c}/\vec{a}]\phi]\!]\rho[g/X, h/Y]\backslash t \end{array}$$

This is proved by applying the inductive hypothesis on $[\vec{c}/\vec{a}]\phi$. □

Definition 2 Two (closed) processes p, q are logically equivalent, and we write $p \sim_{\mathcal{L}} q$, if they cannot be distinguished by a (closed) formula. Formally:

$$p \sim_{\mathcal{L}} q \quad \text{iff} \quad \forall \phi\, (\models p : \phi \ \text{ iff } \ \models q : \phi)$$

Proposition 4 *If $p \sim q$ then $p \sim_{\mathcal{L}} q$.*

PROOF. Least and greatest fixed points can be expressed as infinite disjunctions and conjunctions, respectively. Proceed then by induction on the structure of the (infinitary) formula. □

Next, we show that there is a *finite* formula that characterizes a process up to $\sim^k$, for $k \in \omega$. To this end, for any process p and $k \in \omega$ we define a formula $C^k(p)$ as follows. We set $C^0(p) = \top$ and

$$C^{k+1}(p) = \bigwedge_{\substack{\alpha \in act(p) \\ p \xrightarrow{\alpha} p'}} \langle\alpha\rangle C^k(p') \ \wedge \bigwedge_{\alpha \in act(p) \cup \{\tau\}} [\alpha](\bigvee_{p \xrightarrow{\alpha} p'} C^k(p')) \ \wedge \ [pact(p)^c]\bot \qquad (2)$$

To prove that $C^k(p)$ characterizes p up to $\sim^k$ we need to observe a peculiar property of the labelled transition system (this property relates to the notion of active name introduced in [MP95].)

Definition 3 The set $Unobs(c, k)$ is composed of those processes such that no sequence of transitions of length at most k has c as a free name. More precisely:

$$\begin{array}{lll} Unobs(c,k) & = \{q \mid & (q \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_h} q_h \xrightarrow{\alpha} q',\ h < k, c \notin n(\alpha_i), \\ & & \alpha_i, \alpha \text{ not free input actions } (i = 1, \ldots, h)) \\ & & \text{implies } c \notin fn(\alpha)\} \end{array}$$

Lemma 1 *If $q \in Unobs(c, k)$ then $q \sim^k \nu c.q$.*

PROOF HINT. We relate the transitions of q and $\nu c.q$. If c plays a role in a labelled reduction then we can observe a transition depending on c, hence contradicting $q \in Unobs(c, k)$. □

Remark 1 The previous lemma fails if we allow matching in the calculus. For instance $p \equiv a(b).(\tau.\tau + [b = c].\tau) \in Unobs(c, 3)$ but $p \sim^3 \nu c.p$ does not hold. A related example shows that theorem 1 fails in the presence of matching, more precisely $\models q : C^k(p)$ does not imply $q \sim^k p$. On the other hand corollary 1 still holds. It would be possible to extend the logic so as to capture the behaviour of processes with matching. Roughly we need a modality that specifies the behaviour of a process on a cofinite collection of input names. If s denotes a finite collection of names then we would introduce modalities $\langle a(b \notin s)\rangle\phi$ and $[a(b \notin s)]\phi$ with the following realizations:

$$\begin{array}{ll} \models p : \langle a(b \notin s)\rangle\phi & \text{if } \exists p', c\,(p \xrightarrow{ac} p' \text{ and } c \notin s \text{ and } \models p' : [c/b]\phi) \\ \models p : [a(b \notin s)]\phi & \text{if } \forall p', c\,(p \xrightarrow{ac} p' \text{ and } c \notin s \text{ implies } \models p' : [c/b]\phi) \end{array}$$

We refrain from going into this development because we want to keep the logic simple and because it is debatable whether matching belongs to the π-calculus.

Lemma 2 *If $\models q : C^k(p)$ and $c \in fn(q)\backslash fn(p)$ then $q \in Unobs(c, k)$.*

PROOF. By contradiction, if $q \notin Unobs(c, k)$ then we have a sequence of transitions $q \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_h} q_h \xrightarrow{\alpha} \cdot$ such that $h < k$, $c \notin n(\alpha_i)$, α_i, α not free input actions ($i = 1, \ldots, h$) and $c \in fn(\alpha)$. Then we can find a sequence of transitions $p \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_h} p_h$ such that $\models q_h : C^{k-h}(p_h)$. Since the transitions α_i do not refer to c it has to be the case that $c \notin fn(p_h)$. But then the formula $[pact(p_h)^c]\bot$ forbids the action α which depends on c, hence contradicting $\models q_h : C^{k-h}(p_h)$. □

Theorem 1 *For any process p and $k \in \omega$: (1) $\models p : C^k(p)$, and (2) $\models q : C^k(p)$ iff $p \sim^k q$.*

PROOF. Statement (1) and direction ($\Leftarrow$) of (2) are easily proven by induction on k. To prove direction ($\Rightarrow$) of (2), suppose $fn(q)\backslash fn(p) = \{\vec{c}\}$. By applying lemma 2 we can conclude that $\nu\vec{c}.q \sim^k q$. It is easy to show that $\models q : C^k(p)$ iff $\models \nu\vec{c}.q : C^k(p)$ (cf. proposition 4). Since $\sim^k$ is transitive, we can now prove the statement by induction on k, supposing that $fn(q) \subseteq fn(p)$. □

Corollary 1 *Let p, q be processes. Then: $p \sim q$ iff $p \sim_{\mathcal{L}} q$.*

PROOF. By proposition 4, if $p \sim q$ then $p \sim_{\mathcal{L}} q$. Vice versa, suppose $p \sim_{\mathcal{L}} q$. From proposition 1(1) we know that for each $k \in \omega$, $\models p : C^k(p)$. Hence for each $k \in \omega$, $\models q : C^k(p)$, which implies $p \sim^k q$, for each $k \in \omega$. From theorem 1(2) we conclude that $p \sim q$. □

4 Model checker

In figure 3 we define a proof system to derive judgments $p : \phi$, where $FV(p) = \emptyset$ and ϕ is a closed formula. We write $\vdash p : \phi$ if the judgment is provable in the system. The rules follow quite closely the formula interpretation defined in figure 2, with the exception of fixed points where we exploit proposition 2.

$$(\top)\quad \frac{\cdot}{p : \top} \qquad (\wedge)\quad \frac{p : \phi \quad p : \psi}{p : \phi \wedge \psi}$$

$$(\vee_l)\quad \frac{p : \phi}{p : \phi \vee \psi} \qquad (\vee_r)\quad \frac{p : \psi}{p : \phi \vee \psi}$$

$$(\equiv)\quad \frac{\phi \equiv \phi' \quad p : \phi'}{p : \phi} \qquad (\langle\_\rangle)\quad \frac{p' : \psi}{p : \langle\alpha\rangle\psi} \text{ for some } p \xrightarrow{\alpha} p'$$

$$([\gamma])\quad \frac{pact(p) \subseteq \{\beta_1, \ldots, \beta_n\}}{p : [\{\beta_1, \ldots, \beta_n\}^c]\bot} \qquad ([\_])\quad \frac{p' : \psi}{p : [\alpha]\psi} \text{ whenever } p \xrightarrow{\alpha} p'$$

$$(\nu)\quad \frac{\vec{b} \Rightarrow p \in t}{p : (\nu X(\vec{a}), t.\phi)(\vec{b})} \qquad (\sigma)\quad \frac{\vec{b} \Rightarrow p \notin t \quad p : [\sigma X(\vec{a}), t \cup \{\vec{b} \Rightarrow p\}.\phi/X, \vec{b}/\vec{a}]\phi}{p : (\sigma X(\vec{a}), t.\phi)(\vec{b})}$$

Figure 3: Model checker

Proposition 5 (soundness) *If* $\vdash p : \phi$ *then* $p \in [\![\phi]\!]$.

Proof. The soundness of the rules concerning recursive formulas is proven by applying proposition 2. For instance, suppose $\models p : [\mu X(\vec{a}), t \cup \{\vec{b} \Rightarrow p\}.\phi/X, \vec{b}/\vec{a}]\phi$ and $\vec{b} \Rightarrow p \notin t$. Apply proposition 2.1 with $F(f)(\vec{b}) = [\![[\vec{b}/\vec{a}]\phi]\!][f/X]\backslash t(\vec{b})$. Then:

$$\begin{array}{lll} \models p : (\mu X(\vec{a}), t.\phi)(\vec{b}) & \text{iff} & \models p : lfp(F)(\vec{b}) \\ & \text{iff} & \models p : [\![[\vec{b}/\vec{a}]\phi]\!][lfp(\lambda f.(F\, f \backslash \vec{b} \Rightarrow p))/X] \end{array}$$

□

Remark 2 1. Note that the following judgment *cannot* be proved:

$$p : (\mu X(\vec{a}), t.\phi)(\vec{b}) \quad \text{whenever } \vec{b} \Rightarrow p \in t$$

This is in accord with (3) in lemma 2. In the implementation of the model checker judgments with this shape correspond to halting conditions.

2. The rule (σ) expresses sufficient and *necessary* conditions, as exemplified in (1-2) of lemma 2. This fact plays a crucial role in proving the completeness of the model checker w.r.t. certain classes of processes. As a rule of thumb, completeness means that we can bound the size of the proof of a judgment. Technically, this amounts to show that a certain rewriting system terminates.

$$(\equiv) \quad \frac{rnm(p,(\alpha)\phi) = d \quad bn(\alpha) = \{c\} \quad p : ([d/c]\alpha)[d/c]\phi}{p : (\alpha)\phi}$$

$$(\langle\_\rangle) \quad \frac{rnm(p,\langle\alpha\rangle\phi) = \emptyset \quad p' : \phi}{p : \langle\alpha\rangle\phi} \text{ for some } p \stackrel{\alpha}{\rightarrow}_N p'$$

$$([\_]) \quad \frac{rnm(p,[\alpha]\phi) = \emptyset \quad p' : \phi}{p : [\alpha]\phi} \text{ whenever } p \stackrel{\alpha}{\rightarrow}_N p'$$

Figure 4: Optimized rules for the model checker

As a first step towards completeness we seek a better control on α-renaming (rule $(\equiv)$) and on the introduction of new free names (rules $(\langle\_\rangle)$ and $([\_])$). To this end we introduce a renaming function rnm defined as follows:

$$rnm(p,(\alpha)\phi) = \begin{cases} d & \text{if } bn(\alpha) = \{c\},\ fst(Ch\backslash(fn(p) \cup fn((\alpha)\phi))) = d,\ d \neq c \\ \emptyset & \text{otherwise} \end{cases}$$

The intuition is as follows: the only point in which we might need to rename in order to apply a (real) rule is in a judgment $p : (\alpha)\phi$. This may happen if the α contains a bound name which clashes with a name free in p. We also observe that the bound name may become free in the premise of the rule. We should be conservative in choosing this new name, for this reason in the definition of rnm we pick up the first name which is not currently used. Next we try to bound the collection of processes that might be considered in the proof. We need some notation.

Definition 4 1. A normalization function $N : Pr \rightarrow Pr$ is a total recursive function such that $p \sim N(p)$, for all processes p, and $fn(p) \supseteq fn(N(p))$.

2. Given a normalization function N, we write $p \stackrel{\alpha}{\rightarrow}_N p'$ if there exists p'' such that $p \stackrel{\alpha}{\rightarrow} p''$ and $N(p'') \equiv p'$.

In practice the role of the normalization function is to collect some garbage such as: restrictions on names that do not occur in the body of the process, terminated processes, obviously deadlocked processes, ... We introduce in figure 4 the "optimized" version of the rules $(\equiv)$, $(\langle\_\rangle)$ and $([\_])$.

Proposition 6 *The optimized rules displayed in figure 4 are sound.*

PROOF. The rule ($\equiv$) is a particular case of the rule allowing for the renaming of formulas. The manipulations operated by the normalization function in the rules ($\langle\_\rangle$) and ($[\_]$) are sound because $p \sim N(p)$ and theorem 1 applies. □

In proving completeness we concentrate on processes that have a finite reachability property.

Definition 5 Let N be a normalization function, and let C be a collection of channels forming an initial segment of Ch. A process p has the C, N-finite reachability property if (when C, N can be derived from the context we say FR-process for a process that has the C, N-finite reachability property):

1. The following set is finite:

$$Red(p, C) = \{p_n \mid p \equiv p_1 \stackrel{\alpha_1}{\rightarrow}_N \cdots \stackrel{\alpha_{n-1}}{\rightarrow}_N p_n \text{ and } n(\alpha_i) \subseteq C \text{ and } n \geq 1\} \quad (3)$$

2. If $q \in Red(p, C)$ then $fn(q) \subset C$.

The sense of the second condition is that whenever we need a fresh name for performing a bound input/output we can find it in C.

Example 1 Consider processes with the shape: $p \equiv \nu c_1 \ldots c_n.(q_1 \mid \cdots \mid q_m)$ where $n \geq 0$, $m \geq 1$, and parallel composition cannot occur in q_i. This is a generalization of the standard idea of considering products of regular processes. Let the normalization function N transform all subterms of the shape $\nu c.p'$ into p', whenever $c \notin fn(p')$. Let C be an initial segment of names containing all $fn(p)$ and exceeding the size of p in the sense that by rewriting p we always have less than $\sharp C$ distinct names. Then we can show that terms with the shape above have the C,N-finite reachability property.

Lemma 3 *Let p have the C, N-finite reachability property, and ϕ be a closed formula. Then we can find an initial segment of channel names C' such that $fn(p) \cup fn(\phi), C \subseteq C'$ satisfying the following property: all names free in a judgment $p' : \phi'$ occurring in a proof (in the optimized system) of the judgment $p : \phi$ belong to C'.*

PROOF. Given a formula ϕ one can bound the number of distinct names that can occur in a formula ϕ' by unfolding and projecting on subformulas. Let this bound be n_ϕ. Then we take $\sharp C' = \sharp C + n_\phi$. □

Remark 3 In the hypotheses of the lemma 3 above it is possible to bound the collection of step functions that can occur in a proof. Let n be the maximal arity of a formula identifier occurring in ϕ. Let C' be the set of names obtained by lemma 3. Let $\sharp Red(p, C')$ be the cardinality of the set of processes that can be reached by normalized reduction starting from the process p (cf. equation 3). Then there can be at most $bnstep(p, \phi, C')$ step functions that occur in the proof, where:

$$bnstep(p, \phi, C') = (\sharp(C'^n)) \cdot \sharp Red(p, C') \quad (4)$$

Let us now outline an argument that shows that when developing a proof bottom up in the hypotheses of lemma 3 we eventually stop. We observe that all rules but (σ) and $(\equiv)$ either entail termination or shrink the size of the formula to be proved. We also note that the optimized rule $(\equiv)$ can be applied at most once consecutively, hence any infinite backward development must include an infinite number of applications of the rule (σ). Now we remark that the rule (σ) adds new step functions to the tags. Since we know that the collection of reachable step functions is finite (cf. equation 4) we might conjecture that this process eventually terminates. We prove this in two steps:

1. We present a simple rewriting system whose strong normalization proof exposes the kernel of the combinatorial problem.

2. We show termination of the bottom up proof development by exhibiting a reduction preserving translation from judgments to terms of the simple rewriting system.

Definition 6 We define a collection of σ-terms as follows:

$$\theta ::= X \mid 1 \mid \theta * \theta \mid \bullet\, \theta \mid \sigma^n X.\theta \qquad (n \in \omega)$$

A term θ can be reduced according to the following rules (the rules are applied at top level only).

$$\begin{array}{ll} \sigma^{n+1} X.\theta \to [\sigma^n X.\theta/X]\theta & \bullet\theta \to \theta \\ \theta * \theta' \to \theta & \theta * \theta' \to \theta' \end{array}$$

Proposition 7 *The rewriting system defined in 6 is strongly normalizing, that is all reduction sequences terminate.*

Proof. Let SN be the collection of strongly normalizing σ-terms. If $\theta \in SN$ let $d(\theta)$ be the lenght of the longest reduction sequence (this is well defined because the reduction tree is finitely branching). We want to prove:

$$\theta, \theta' \in SN \quad \Rightarrow \quad [\theta'/X]\theta \in SN \tag{5}$$

We prove (5) by induction on $d(\theta)$. [2] The only interesting case is when θ has the shape $\sigma^{n+1} Y.\theta$. Then we observe:

$$\begin{array}{ll} [\theta'/X](\sigma^{n+1} Y.\theta) & \equiv \sigma^{n+1} Y.[\theta'/X]\theta \\ \to [\sigma^n Y.[\theta'/X]\theta/Y]([\theta'/X]\theta) & \equiv [\theta'/X][\sigma^n Y.\theta/Y]\theta \end{array}$$

We note that $(\sigma^{n+1} Y.\theta) \to [\sigma^n Y.\theta/Y]\theta \in SN$. Hence, we can apply the inductive hypothesis on $d([\sigma^n Y.\theta/Y]\theta)$, and we conclude that $[\theta'/X][\sigma^n Y.\theta/Y]\theta \in SN$.

[2] This proof mimicks proof techniques which show the strong normalization of the simply typed and labelled λ-calculus (cf. [VD80], chapter IV). The proof is *elementary* in the sense that it can be formalized in Peano arithmetic, in particular no reducibility predicate is required.

Next we prove that all σ-terms are strongly normalizing. We proceed by induction on a relation $\succ$ which is the least transitive relation such that:

$$\begin{array}{ll} \sigma^{n+1}X.\theta \succ \sigma^n X.\theta & \sigma^{n+1}X.\theta \succ \theta \quad \bullet\theta \succ \theta \\ \theta * \theta' \succ \theta & \theta * \theta' \succ \theta' \end{array}$$

Clearly $\succ$ is a well founded relation. Again the only interesting case is when the term has the shape $\sigma Y^{n+1}.\theta$. By the inductive hypothesis $\sigma Y^n.\theta \in SN, \theta \in SN$, and by (5) $[\sigma Y^n.\theta/Y]\theta \in SN$. □

Lemma 4 *Consider a bottom up proof development starting from a root $p : \phi$, where p is an FR-process. There is a translation $\langle\_\rangle$ from formulas to σ-terms which preserves reductions, that is if $p'' : \phi''$ is a premise of $p' : \phi'$ in the proof development (by application of a rule different from $(\equiv)$, otherwise $\langle\phi'\rangle = \langle\phi''\rangle$) then $\langle\phi'\rangle \rightarrow \langle\phi''\rangle$. Consequently, all paths in the proof tree having root $p : \phi$ are finite.*

Proof. Let $s = bnstep(p, \phi, C')$ be as in equation 4. Then s exceeds the cardinality of all tags that may occur by unfolding the fixed points. We associate a σ-term to a formula as follows:

$$\begin{array}{llll} \langle\top\rangle & = 1 & \langle\bot\rangle & = 1 \\ \langle\phi \wedge \psi\rangle & = \langle\phi\rangle * \langle\psi\rangle & \langle\phi \vee \psi\rangle & = \langle\phi\rangle * \langle\psi\rangle \\ \langle(\alpha)\phi\rangle & = \bullet\langle\phi\rangle & \langle X(\vec{a})\rangle & = X \\ \langle(\sigma X(\vec{a}), t.\phi)(\vec{b})\rangle & = \sigma X^{(s-\sharp t)}.\langle\phi\rangle & & \end{array}$$

Suppose $p'' : \phi''$ is a premise of $p' : \phi'$ in the proof developement (by application of a rule different from $(\equiv)$). We show $\langle\phi'\rangle \rightarrow \langle\phi''\rangle$ by inspection of the proof rules. The only interesting case is the (σ) rule. Since in the translation we have picked $s = bnstep(p, \phi, C')$ bigger than $\sharp t$ we can compute:

$$\begin{array}{ll} \langle(\sigma X(\vec{a}), t.\phi)(\vec{b})\rangle & = \sigma X^{(s-\sharp t)}.\langle\phi\rangle \\ \rightarrow [\sigma X^{(s-\sharp t-1)}.\langle\phi\rangle/X]\langle\phi\rangle & = \langle[\sigma X(\vec{a}), t \cup \{\vec{b} \Rightarrow p\}.\phi/X, \vec{b}/\vec{a}]\phi\rangle \end{array}$$

If there was an infinite path in the proof tree then there would be an infinite reduction from the σ-term $\langle p : \phi\rangle$, contradicting proposition 7. □

Theorem 2 *The model checker is complete on FR-processes.*

Proof. We show by induction on ϕ that $\models p : \phi$ iff a proof rule applies. Lemma 4 bounds the depth of a path in a bottom up proof development. Hence, if $\models p : \phi$ by developing the proof bottom up we eventually obtain a proof of $p : \phi$. □

Moreover the logic is sufficiently expressive to characterize FR-processes up to bisimulation equivalence.

Theorem 3 *Let p be a processes having the C,N-finite reachability property. Then there is a formula $C(p)$ involving only greatest fixed points such that for any process q, $\models q : C(p)$ iff $q \sim p$.*

PROOF. Suppose $Red(p,C) = \{p_1, \ldots, p_n\}$. We consider the system of equations $X_i(C) = \phi_i$, for $i = 1, \ldots, n$, where:

$$\phi_i \equiv \bigwedge_{\substack{\alpha \in act(p_i) \\ p_i \stackrel{\alpha}{\rightarrow} p_j}} \langle\alpha\rangle X_j(C) \ \wedge \bigwedge_{\alpha \in act(p_i) \cup \{\tau\}} [\alpha](\bigvee_{p_i \stackrel{\alpha}{\rightarrow} p_j} X_j(C)) \ \wedge \ [pact(p_i)^c]\bot \qquad (6)$$

When writing $X_i(C)$ we intend that the collection of names in C has been ordered in a list to form the parameters of the formula X_i. For the sake of brevity we omit to write these parameters in the following as all the formula variables depend on the same list of parameters. [3]

1. We show that there are formulas $\psi_1, \ldots, \psi_n$ such that:

$$\psi_i = [\psi_1/X_1, \ldots, \psi_n/X_n]\phi_i \quad (i = 1, \ldots, n) \qquad (7)$$

   where equality can be proven simply by unfolding fixed points. We proceed by induction on n. If $n = 1$ let $\psi_1 = \nu X_1.\phi_1$. If $n > 1$ let $\psi'_n = \nu X_n.\phi_n$. Consider the system:

$$X_i = [\psi'_n/X_n]\phi_i \quad (i = 1, \ldots, n \perp 1)$$

   which by inductive hypothesis has a solution $\psi_1, \ldots, \psi_{n-1}$ satisfying:

$$\psi_i = [\psi_1/X_1, \ldots, \psi_{n-1}/X_{n-1}][\psi'_n/X_n]\phi_i \quad (i = 1, \ldots, n \perp 1)$$

   Then, taking $\psi_n = [\psi_1/X_1, \ldots, \psi_{n-1}/X_{n-1}]\psi'_n$, we observe that $\psi_1, \ldots, \psi_n$ is a solution for the system (7).

2. Having obtained an explicit solution of the system (7) we can prove by induction on $k \in \omega$ that $\models q : \psi_i$ implies $q \sim^k p_i$ (cf. proof theorem 1). Hence $\models q : \psi_i$ implies $q \sim p_i$.

3. We can associate a monotone functional $\lambda(X_1, \ldots, X_n).(\phi_1, \ldots, \phi_n)$ to the system (7). We can show that the solution $\psi_1, \ldots, \psi_n$ built in (1) is the greatest fixed point of this functional.

4. Next we prove $\models p_i : \psi_i$. From this we can derive that if $q \sim p_i$ then $\models q : \psi_i$, as bisimilar processes cannot be distinguished by a formula. To this end we use a greatest fixed point reasoning principle applied to the system of equations (7):

$$\frac{\models p_i : X_i \ \ (i = 1, \ldots, n) \text{ implies } \models p_i : \phi_i \ \ (i = 1, \ldots, n)}{\models p_i : \nu(\lambda(X_1, \ldots, X_n).(\phi_1, \ldots, \phi_n))_i \equiv \psi_i \ \ (i = 1, \ldots, n)}$$

□

[3] This way of deriving the characteristic formula is particularly honerous. In practice, if the process p is specified by a system of parametric process equations using prefix and sum, then it is possible to build a system of parametric formula equations having a comparable size.

5 Proof system

We develop a proof system supporting the compositional proof of process properties. Following [AD95, Dam95] basic judgments of such a proof system take the form $\Gamma \vdash p : \phi$ where Γ is a sequence of the form $x_1 : \phi_1, \ldots, x_n : \phi_n$. Such a judgment states properties ϕ of p relative to properties ϕ_i of its components, x_i. In the case of the π-calculus a main issue is how to cater for private names, in particular name generation and scope extrusion. For instance one will wish to verify properties of a process $\nu a.p(x)$ relative to a property, say ψ, of x. In general ψ must be allowed to depend on a. Thus, the scope of a needs to be extended to cover also ψ. For this purpose we work instead with judgments of the form $\Gamma \vdash^s p : \phi$ where s is a finite set of restricted channel names whose scope extend over Γ and p.

Judgments. Let s range over finite sets of channels. Write s, a for $s \cup \{a\}$ and $s \perp a$ for $s \backslash \{a\}$, and if $s = \{a_1, \ldots, a_m\}$ then $\nu s.p = \nu a_1 \ldots \nu a_m.p$. A *basic judgment* is an expression of the form

$$x_1 : \psi_1, \ldots, x_n : \psi_n \vdash^s p : \psi \tag{8}$$

satisfying the properties:

1. All x_i are distinct. The order of hypotheses in the context is irrelevant.
2. ψ_i, $1 \leq i \leq n$, and ψ are closed.
3. $FV(p) \subseteq \{x_1, \ldots, x_n\}$.
4. $fn(\psi) \cap s = \emptyset$.

The judgment (8) is interpreted as follows:

For all $p_1, \ldots, p_n$, if $\models p_i : \psi_i$ for all i $(1 \leq i \leq n)$ then $\models \nu s.[\vec{p}/\vec{x}]p : \psi$.

We write $\Gamma \models^s p : \psi$ if $\Gamma \vdash^s p : \psi$ is valid in this sense.

The proof system. In the following we present a proof system for deriving valid judgments following [AD95, Dam95]. The rules are divided into three groups: one depending on the logical structure of the formula (figure 5), one depending on both formula structure and process structure (figures 6, 7, and 8), and finally a group of ancillary rules including those that depend on the structure of process terms only (figure 9). We omit symmetric rules for conjunction, disjunction, sum, and parallel composition, in practice these operators should be regarded as commutative.

For the rules we suppose that the premises are well formed judgments, and the conclusion judgment of each rule is relevant only if the judgment is well-formed. The rules use the abbreviation *a fresh*. In the context of a single judgment

$$\bot \quad \frac{\cdot}{\Gamma, x : \bot \vdash^s p : \phi} \qquad \top \quad \frac{\cdot}{\Gamma \vdash^s p : \top}$$

$$\wedge_L \quad \frac{\Gamma, x : \psi_1 \vdash^s p : \phi}{\Gamma, x : \psi_1 \wedge \psi_2 \vdash^s p : \phi} \qquad \wedge_R \quad \frac{\Gamma \vdash^s p : \phi_1 \quad \Gamma \vdash^s p : \phi_2}{\Gamma \vdash^s p : \phi_1 \wedge \phi_2}$$

$$\vee_L \quad \frac{\Gamma, x : \psi_1 \vdash^s p : \phi \quad \Gamma, x : \psi_2 \vdash^s p : \phi}{\Gamma, x : \psi_1 \vee \psi_2 \vdash^s p : \phi} \qquad \vee_R \quad \frac{\Gamma \vdash^s p : \phi_1}{\Gamma \vdash^s p : \phi_1 \vee \phi_2}$$

$$(cut) \quad \frac{\Gamma \vdash^{\emptyset} q : \psi \quad \Gamma, x : \psi \vdash^s p : \phi}{\Gamma \vdash^s [q/x]p : \phi}$$

Figure 5: Logical rules

$$(0) \quad \frac{\cdot}{\Gamma \vdash^s 0 : [\alpha]\phi}$$

$$(out^1) \quad \frac{\Gamma \vdash^s p : \phi}{\Gamma \vdash^s \overline{a}b.p : (\overline{a}b)\phi} \qquad (out^2) \quad \frac{prj(\alpha) \neq \overline{a}}{\Gamma \vdash^s \overline{a}b.p : [\alpha]\phi}$$

$$(out^1_{ex}) \quad \frac{\Gamma \vdash^s p : \phi \quad b \notin s}{\Gamma \vdash^{s,b} \overline{a}b.p : (\overline{a}(b))\phi} \qquad (in^1) \quad \frac{\Gamma \vdash^s [c/b]p : \phi}{\Gamma \vdash^s a(b).p : (ac)\phi}$$

$$(in^2) \quad \frac{prj(\alpha) \neq a}{\Gamma \vdash^s a(b).p : [\alpha]\phi} \qquad (in^1_{ex}) \quad \frac{\Gamma \vdash^s p : \phi}{\Gamma \vdash^s a(b).p : (a(b))\phi} \quad (b \text{ fresh})$$

$$(+^{\langle\,\rangle}) \quad \frac{\Gamma \vdash^s p : \langle\alpha\rangle\phi}{\Gamma \vdash^s p + q : \langle\alpha\rangle\phi} \qquad (+^{[\,]}) \quad \frac{\Gamma \vdash^s p : [\alpha]\phi \quad \Gamma \vdash^s q : [\alpha]\phi}{\Gamma \vdash^s p + q : [\alpha]\phi}$$

Figure 6: Process structure rules (minus parallel composition)

$$(sync^{\langle\,\rangle}) \quad \frac{\Gamma, x:\psi_1, y:\psi_2 \vdash^s x \mid y:\phi}{\Gamma, x:\langle ab\rangle\psi_1, y:\langle \overline{a}b\rangle\psi_2 \vdash^s x \mid y:\langle\tau\rangle\phi}$$

$$(sync_{ex}^{\langle\,\rangle}) \quad \frac{\Gamma, x:\psi_1, y:\psi_2 \vdash^{s,b} x \mid y:\phi}{\Gamma, x:\langle a(b)\rangle\psi_1, y:\langle \overline{a}(b)\rangle\psi_2 \vdash^s x \mid y:\langle\tau\rangle\phi}$$

$$(comp_1^{\langle\,\rangle}) \quad \frac{\Gamma, x:\psi_1, y:\psi_2 \vdash^s x \mid y:\phi \quad n(\alpha)\cap s = \emptyset}{\Gamma, x:\langle\alpha\rangle\psi_1, y:\psi_2 \vdash^s x \mid y:\langle\alpha\rangle\phi}$$

$$(comp_{1,ex}^{\langle\,\rangle}) \quad \frac{\Gamma, x:\psi_1, y:\psi_2 \vdash^s x \mid y:\phi \quad a,b \notin s}{\Gamma, x:\langle \overline{a}b\rangle\psi_1, y:\psi_2 \vdash^{s,b} x \mid y:\langle \overline{a}(b)\rangle\phi}$$

$$(comp^{[\,]}) \quad \frac{\begin{array}{c}\Gamma, x:\psi_{1,1}, y:\psi_{2,2} \vdash^s x \mid y:\psi \\ \Gamma, x:\psi_{1,2}, y:\psi_{2,1} \vdash^s x \mid y:\psi \\ \alpha \neq \tau \quad n(\alpha)\cap s = \emptyset\end{array}}{\Gamma, x:\psi_{1,1}\wedge[\alpha]\psi_{1,2}, y:\psi_{2,1}\wedge[\alpha]\psi_{2,2} \vdash^s x \mid y:[\alpha]\psi}$$

$$(comp_{ex}^{[\,]}) \quad \frac{\begin{array}{c}\Gamma, x:\psi_{1,1}, y:\psi_{2,2} \vdash^s x \mid y:\psi \\ \Gamma, x:\psi_{1,2}, y:\psi_{2,1} \vdash^s x \mid y:\psi \\ \alpha \neq \tau \quad a,b \notin s\end{array}}{\Gamma, x:\psi_{1,1}\wedge[\overline{a}b]\psi_{1,2}, y:\psi_{2,1}\wedge[\overline{a}b]\psi_{2,2} \vdash^{s,b} x \mid y:[\overline{a}(b)]\psi}$$

$$(sync^{[\,]}) \quad \frac{\begin{array}{c}\Gamma, x:\psi_{1,1}, y:\psi_2 \vdash^s x \mid y:\psi \\ \Gamma, x:\psi_1, y:\psi_{2,1} \vdash^s x \mid y:\psi \\ \Gamma, x:\psi_{1,2}^{\overline{a}b}, y:\psi_{2,3}^{\overline{a}b} \vdash^s x \mid y:\psi \quad (\forall \overline{a}b \in \gamma_1^c \cup \gamma_2^c) \\ \Gamma, x:\psi_{1,3}^{\overline{a}b}, y:\psi_{2,2}^{\overline{a}b} \vdash^s x \mid y:\psi \quad (\forall \overline{a}b \in \gamma_1^c \cup \gamma_2^c) \\ \Gamma, x:\psi_{1,4}^{\overline{a}}, y:\psi_{2,5}^{\overline{a}} \vdash^{s,b} x \mid y:\psi \quad (\forall \overline{a} \in \gamma_1^c \cup \gamma_2^c) \\ \Gamma, x:\psi_{1,5}^{\overline{a}}, y:\psi_{2,4}^{\overline{a}} \vdash^{s,b} x \mid y:\psi \quad (\forall \overline{a} \in \gamma_1^c \cup \gamma_2^c)\end{array}}{\Gamma, x:\psi_1', y:\psi_2' \vdash^{s-b} x \mid y:[\tau]\psi}$$

where, for $i \in \{1,2\}$,

$$\psi_i' = \psi_i \wedge [\gamma_i]\bot \wedge [\tau]\psi_{i,1} \wedge \Big(\bigwedge_{\overline{a}b \in \gamma_1^c \cup \gamma_2^c} [\overline{a}b]\psi_{i,2}^{\overline{a}b} \wedge [ab]\psi_{i,3}^{\overline{a}b}\Big) \wedge \Big(\bigwedge_{\overline{a} \in \gamma_1^c \cup \gamma_2^c} [\overline{a}(b)]\psi_{i,4}^{\overline{a}} \wedge [a(b)]\psi_{i,5}^{\overline{a}}\Big)$$

Figure 7: Process structure rules (parallel composition)

$$(\hat{0}) \quad \frac{\cdot}{\Gamma \vdash^s 0 : [\gamma]\bot} \qquad (\hat{out}^2) \quad \frac{\overline{a}b \notin \gamma \text{ or } a \in s}{\Gamma \vdash^s \overline{a}b.p : [\gamma]\bot}$$

$$(\hat{out}^2_{ex}) \quad \frac{\overline{a} \notin \gamma \text{ or } a \in s}{\Gamma \vdash^{s,b} \overline{a}b.p : [\gamma]\bot} \qquad (\hat{in}^2) \quad \frac{a \notin \gamma \text{ or } a \in s}{\Gamma \vdash^s a(b).p : [\gamma]\bot}$$

$$(\hat{+}^{[\,]}) \quad \frac{\Gamma \vdash^s p : [\gamma]\bot \;\; \Gamma \vdash^s q : [\gamma]\bot}{\Gamma \vdash^s p + q : [\gamma]\bot} \qquad (\hat{comp}^{[\,]}) \quad \frac{\Gamma \vdash^s p_1 : [\gamma]\bot \;\; \Gamma \vdash^s p_2 : [\gamma]\bot}{\Gamma \vdash^s p_1 \mid p_2 : [\gamma]\bot}$$

Figure 8: Process structure rules, generalised box

$$(\nu) \quad \frac{\Gamma \vdash^{s,a} p : \phi}{\Gamma \vdash^{s-a} \nu a.p : \phi} \;\; (a \text{ fresh}) \qquad (rec) \quad \frac{\Gamma \vdash^s [(recA(\vec{a}).p)/A, \vec{b}/\vec{a}]p : \phi}{\Gamma \vdash^s (recA(\vec{a}).p)(\vec{b}) : \phi}$$

$$(btf) \quad \frac{d \notin fn((a(c))\phi)}{\Gamma, x : (a(c))\phi \vdash x : (ad)[d/c]\phi}$$

$$(mon) \quad \frac{\Gamma, x : \psi \vdash^s x : \phi}{\Gamma, x : (\alpha)\psi \vdash^s x : (\alpha)\phi} \qquad (mon_{ex}) \quad \frac{\Gamma, x : \psi \vdash^{s-b} x : \phi}{\Gamma, x : (\overline{a}b)\psi \vdash^{s,b} x : (\overline{a}(b))\phi}$$

$$(\hat{mon}) \quad \frac{\gamma' \subseteq \gamma}{\Gamma, x : [\gamma]\bot \vdash^s x : [\gamma']\bot} \qquad (\hat{mon}') \quad \frac{prj(\alpha) \in \gamma}{\Gamma, x : [\gamma]\bot \vdash^s x : [\alpha]\phi}$$

Figure 9: Ancillary rules

$\Gamma \vdash^s p : \phi$ this means that a does not occur freely in neither Γ, s, p, nor ϕ, and in the context of a rule

$$\frac{\Gamma' \vdash^{s'} p' : \phi'}{\Gamma \vdash^s p : \phi}$$

the abbreviation means that a is fresh for $\Gamma \vdash^s p : \phi$.

We prove a completeness result to the effect that if $\Gamma \vdash^s p : \phi$ is valid where each formula in Γ is a characteristic formula of some process to a depth at least the modal depth of ϕ, then $\Gamma \vdash^s p : \phi$ is also provable. In order to achieve this, certain relationships between free and bound input/output actions, must be made explicit, see in particular rules $(comp^{\langle\rangle}_{1,ex})$, $(comp^{[]}_{ex})$, and (btf) (btf stands for bound to free).

At a first reading one may concentrate on the fragment including formulas $\phi ::= \top \mid \phi \wedge \phi \mid \langle\alpha\rangle\phi$ and processes $p ::= 0 \mid \overline{a}b.p \mid a(b).p \mid \nu a.p \mid (p \mid p) \mid x$. For proving the (weak) completeness of this fragment one just needs the following 14 rules: $(\top)$, $(\wedge_L)$, $(\wedge_R)$, (cut), (out^1), (out^1_{ex}), (in^1), (in^1_{ex}), $(sync^{\langle\rangle})$, $(sync^{\langle\rangle}_{ex})$, $(comp^{\langle\rangle}_1)$, $(comp^{\langle\rangle}_{1,ex})$, (ν), and (btf).

The precise formulation of the proof rules is open to a lot of variation. For instance, it would be possible to be less explicit in our use of the cut-rule by replacing the rules of figure 7 with rules of the form:

$$\frac{\Gamma \vdash^{\emptyset} p : \langle\alpha\rangle\phi \quad \Gamma, x : \phi \vdash^s x \mid q : \psi \quad n(\alpha) \cap s = \emptyset}{\Gamma \vdash^s p \mid q : \langle\alpha\rangle\phi}$$

The soundness proof relies on the following observation.

Proposition 8 *For all a not free in p or ϕ, $\models p : \phi$ if and only if $\models [a/b]p : [a/b]\phi$.*

Note the following consequence of proposition 8: If a does not occur in p or ϕ and b does not occur in ϕ then $\models p : \phi$ if and only if $\models [a/b]p : \phi$. This allows us to overcome some difficulties in the soundness proof.

Proposition 9 *If $\Gamma \vdash^s p : \phi$ then $\Gamma \models^s p : \phi$.*

PROOF. Let $\Gamma = x_1 : \phi_1, \ldots, x_n : \phi_n$ and let δ be a Γ-*validating substitution*, i.e. a mapping with domain $\{x_1, \ldots, x_n\}$ such that $\models \delta(x_i) : \phi_i$ for each $i : 1 \leq i \leq n$. We go through the rules on by one.

(cut). Assume $\models q\delta : \psi$ and that whenever $\models r : \psi$ then $\models \nu s.p([r/x]\delta) : \phi$. Then $\models \nu s.([q\delta/x]p)\delta : \phi$ and $\models \nu s.([q/x]p)\delta$.

(out^1). Assume $\models \nu s.p\delta : \phi$. It suffices to show $\models \nu s.(\overline{a}b.p)\delta : \langle\overline{a}b\rangle\phi$. By well-formedness we know that $a, b \notin s$. But this is trivial.

$(sync^{\langle\rangle}_{ex})$. Assume that whenever $\models q_1 : \psi_1$ and $\models q_2 : \psi_2$ then $\models \nu s.\nu b.q_1 \mid q_2 : \phi$. Let $\models r_1 : \langle a(b)\rangle\psi_1$ and $\models r_2 : \langle\overline{a}(b)\rangle\psi_2$. Then we find a fresh c such that $r_1 \stackrel{a(c)}{\rightarrow} q_1$ and $\models q_1 : [c/b]\psi_1$, and a fresh d such that $r_2 \stackrel{\overline{a}(d)}{\rightarrow} q_2$ and $\models q_2 : [d/b]\psi_2$.

Then $\models [b/c]q_1 : \psi_1$, $\models [b/d]q_2 : \psi_2$, $\nu s.r_1 \mid r_2 \xrightarrow{\tau} \nu s.\nu b.[b/c]q_1 \mid [b/d]q_2$, and $\models \nu s.\nu b.[b/c]q_1 \mid [b/d]q_2 : \phi$ as desired.

$(comp^{\langle\rangle}_{ex})$. Assume that $a, b \notin s$, and that whenever $\models q_1 : \psi_1$ and $\models q_2 : \psi_2$ then $\models \nu s.q_1 \mid q_2 : \phi$. Suppose that $\models r_1 : \langle \overline{a}b\rangle\psi_1$ and $\models q_2 : \psi_2$. Then $r_1 \xrightarrow{\overline{a}b} q_1$ such that $\models q_1 : \psi_1$. Then $\nu s.\nu b.r_1 \mid q_2 \xrightarrow{\overline{a}(b)} \nu s.q_1 \mid q_2$, and by the assumptions $\models \nu s.q_1 \mid q_2 : \phi$, completing the case.

(ν). For simplicity assume that $n = 1$, that $\delta(x_1) = q$, and that $a \notin s$. Generalising the proof is not hard. Assume $\models \nu s.\nu a.[q/x_1]p : \phi$. Since a is fresh it does not occur freely in ϕ_1. Since $\models q : \phi_1$ and $a \notin fn(\phi_1)$ we can find some fresh b such that $\models [b/a]q : \phi_1$ too (proposition 8). Then $\models \nu s.\nu a.[[b/a]q/x_1]p : \phi$. Since $a \notin fn([b/a]q)$ also $\models \nu s.[[b/a]q/x_1]\nu a.p : \phi$. But since a does not occur in ϕ also $\models [a/b]\nu s.[[b/a]q/x_1]\nu a.p : \phi$, thus, since $a \notin s$, $\models \nu s.[q/x_1]\nu a.p : \phi$ as desired.

(mon). Assume that whenever $\models q : \psi$ then $\models \nu s.q : \phi$. Assume that $\models r : \langle\alpha\rangle\psi$ (the case for $[\alpha]$ is analogous). Let first $\alpha = ab$. Then we find a q such that $r \xrightarrow{ab} q$ and $\models q : \psi$. Then $\models \nu s.q : \phi$ by the assumptions. Moreover, since $a, b \notin s$ by the well-formedness condition, $\nu s.r \xrightarrow{ab} \nu s.q$, so indeed $\models \nu s.r : \langle\alpha\rangle\phi$. The case for $\overline{a}b$ is similar. Let then $\alpha = \overline{a}(b)$. Either b is free in r or it is not (the easier case). So assume $b \in fn(r)$. Since $\models r : \langle\alpha\rangle\psi$ also $\models [c/b]r : [c/b]\langle\alpha\rangle\psi$ where c is fresh by proposition 8. Note that $c \notin fn(\psi)$. Then we find a q such that $[c/b]r \xrightarrow{\overline{a}(b)} q$ and $\models q : \psi$. Then $r \xrightarrow{\overline{a}(d)} [b/c][d/b]q$, so also $\nu s.r \xrightarrow{\overline{a}(d)} [b/c][d/b]\nu s.q$ (assuming w.l.o.g. that $b \notin fn(s)$). Now $\models \nu s.q : \phi$ so $\models [d/b]\nu s.q : [d/b]\phi$ (proposition 8 again), and then, since c is not free in $[d/b]\phi$ we obtain $\models [b/c][d/b]\nu s.q : [d/b]\phi$, completing the case.

(mon_{ex}). Assume that $b \notin s$ and that whenever $\models q : \psi$ then $\models \nu s.q : \phi$. Assume that $\models q : \langle\overline{a}b\rangle\psi$. Then $q \xrightarrow{\overline{a}b} q'$ for some q' such that $\models q' : \psi$. By well-formedness of the conclusion, $a \notin s$. Then $\nu s.\nu b.q \xrightarrow{\overline{a}(b)} \nu s.q'$. By the assumption, $\models \nu s.q' : \phi$. But then we have shown that $\models \nu s.\nu b.q : \langle\overline{a}(b)\rangle\phi$, as required.

The remaining cases are all patterned upon the above and left for the reader. □

We proceed to show completeness. The following lemma is important in the latter case since it allows assumptions of the form $x : C^k(q)$ to be exchanged for substitutions by q.

Lemma 5 *1. For all $k \in \omega$, if $q_i \sim^k q_i'$, for $i = 1, \ldots, n$, then $\nu s.[\vec{q}/\vec{x}]p \sim^k \nu s.[\vec{q'}/\vec{x}]p$.*

2. Let $|\phi| \leq k_i$, for $i = 1, \ldots, n$. Then $x_1 : C^{k_1}(q_1), \ldots, x_n : C^{k_n}(q_n) \models^s p : \phi$ if and only if $\models \nu s.[q_1/x_1, \ldots, q_n/x_n]p : \phi$.

Note that, given that $\models q : C^k(q)$, 5.2 is an immediate consequence of 5.1, the proof of which we omit.

Theorem 4 *Let* $\Gamma = x_1 : C^{k_1}(q_1), \ldots, x_n : C^{k_n}(q_n)$. *If* $\Gamma \models^s p : \phi$ *and* $|\phi| \leq k_i$, *for* $i = 1, \ldots, n$, *then* $\Gamma \vdash^s p : \phi$.

PROOF. First observe that, because process terms are assumed to be guarded, it is a routine matter to lift a proof of 4 for non-recursive processes to include recursion. The proof proceeds by induction on the lexicographic order $(|\phi|, struct(p), struct(\phi))$. Assume $\Gamma \models^s p : \phi$ and $|\phi| \leq k_i$, for $i = 1, \ldots, n$. We proceed by cases on $struct(\phi)$.

$\phi = \bot$. Contradiction.

$\phi = \top$. Use the rule $\top$.

$\phi = \phi_1 \wedge \phi_2$. We obtain $\Gamma \models^s p : \phi_i$ for $i = 1$ and $i = 2$. By the innermost ind. hyp. (since $|\phi|$ has increased), we obtain $\Gamma \vdash^s p : \phi_i$ for $i = 1$ and $i = 2$. Then, by $\wedge_R$, $\Gamma \vdash^s p : \phi$.

$\phi = \phi_1 \vee \phi_2$. Observe that $\models \nu s.[q_1/x_1, \ldots, q_n/x_n]p : \phi$. Then $\models \nu s.[q_1/x_1, \ldots, q_n/x_n]p : \phi_i$ for $i = 1$ or $i = 2$. Then (by lemma 5.2), $\Gamma \models^s p : \phi_i$, for that particular i. Then the conclusion follows by $\vee_R$.

$\phi = [\gamma]\bot$. We proceed by cases on $struct(p)$.

- $p = 0$: $\Gamma \vdash^s p : \phi$ by $(\hat{0})$.
- $p = a(b).p'$: If $\Gamma \models^s p : \phi$ then either $a \in s$ or $a \notin \gamma$. In any case $\Gamma \vdash^s p : \phi$ by $(\hat{in}^2)$.
- $p = \overline{a}b.p'$: The proof is similar to the previous case, except that $(\hat{out}^2)$ or $(\hat{out}^2_{ex})$ are used dependent on whether $b \in s$ or not.
- $p = \nu a.p'$: Suppose $\Gamma \models^s p : \phi$. We can assume that a is not among the names free in $q_1, \ldots, q_n$. By 5.2, $\models \nu s.[q_1/x_1, \ldots, q_n/x_n]\nu a.p' : \phi$, and then $\models \nu s.\nu a.[q_1/x_1, \ldots, q_n/x_n]p' : \phi$. Thus $\Gamma \vdash^{s,a} p' : \phi$. By the second level ind. hyp., $\Gamma \vdash^{s,a} p' : \phi$, and thus we are done by (ν).
- $p = p_1 + p_2$: If $\Gamma \models^s p : \phi$ then $\Gamma \models^s p_i : \phi$ for $i = 1$ and $i = 2$. By the second level ind. hyp., $\Gamma \vdash^s p_i : \phi$, and then by $(\hat{+}^{[]})$ we are done.
- $p = p_1 \mid p_2$: Assume $\Gamma \models^s p : \phi$. Then (since $\tau \notin \gamma$) $\Gamma \models^s p_i : \phi$ for $i = 1$ and $i = 2$, and by the second level ind. hyp. plus $(\hat{comp}^{[]})$ the result follows.
- $p = x_i$: Note that $\Gamma \models^s p : [\gamma]\bot$ iff $\models \nu s.q_i : [\gamma]\bot$. Since, by the convention, $s \cap \gamma = \emptyset$ this is the case if and only if $\models q_i : [\gamma]\bot$, and then $\Gamma \vdash^s p : [\gamma]\bot$ by $(\wedge_L)$ and $(\hat{mon})$.

$\phi = \langle\alpha\rangle\phi'$. We proceed by cases on $struct(p)$.

- $p = 0$: Contradiction.

- $p = a(b).p'$: If $\Gamma \models^s p : \phi$ then α must have one of the form ac or $a(c)$ where $a \notin s$. Suppose first $\alpha = ac$. Then $\Gamma \models^s [c/b]p' : \phi'$ so by the outer ind. hyp., $\Gamma \vdash^s [c/b]p' : \phi'$ giving $\Gamma \vdash^s p : \phi$ by (in^1). Second, if $\alpha = a(b)$ (assuming $b \notin fn(\phi)$, otherwise alpha-conversion is needed) the result is obtained by (in^1_{ex}).

- $p = \overline{a}b.p'$: α must have one of the forms $\overline{a}b$ or $\overline{a}(b)$ (again appealing to alpha-conversion, if needed). In the first case the result is obtained by (out^1) and the second by (out^1_{ex}).

- $p = \nu a.p'$: Use (ν) and the second level ind. hyp..

- $p = p_1 + p_2$: Use lemma 5.2, $(+^{\langle\rangle})$, and the second level ind. hyp..

- $p = p_1 \mid p_2$: If $\Gamma \models^s p : \phi$ then $\models \nu s.[q_1/x_1, \ldots, q_n/x_n](p_1 \mid p_2) : \langle\alpha\rangle\phi'$. α has one of the forms (1) τ, (2) $\overline{a}b$, (3) $\overline{a}(b)$, (4) ab, or (5) $a(b)$. Assume case (1). Either (leaving out symmetrical subcases) (i) $[\vec{q}/\vec{x}]p_1 \stackrel{\tau}{\rightarrow} p_1'$ and $\models \nu s.p_1' \mid [\vec{q}/\vec{x}]p_2 : \phi'$ or (ii) for some a, b, $[\vec{q}/\vec{x}]p_1 \stackrel{\overline{a}b}{\rightarrow} p_1'$, $[\vec{q}/\vec{x}]p_2 \stackrel{ab}{\rightarrow} p_2'$, and $\models \nu s.p_1' \mid p_2' : \phi'$, or (iii) for some a, b, $[\vec{q}/\vec{x}]p_1 \stackrel{\overline{a}(b)}{\rightarrow} p_1'$, $[\vec{q}/\vec{x}]p_2 \stackrel{a(b)}{\rightarrow} p_2'$, and $\models \nu s.\nu b.p_1' \mid p_2' : \phi'$. We take each subcase in turn.

  (i) By lemma 5.2 and the second level ind. hyp. we obtain $\Gamma \vdash p_1 : \langle\tau\rangle C^{k-1}(p_1')$, and (by the outer ind. hyp.) $x_0 : C^{k-1}(p_1'), x_1 : C^{k_1}(q_1), \ldots, x_n : C^{k_n}(q_n) \vdash^s x_0 \mid p_2 : \phi'$. Then, using $(comp_1^{\langle\rangle})$, $x_0 : \langle\tau\rangle C^{k-1}(p_1'), x_1 : C^{k_1}(q_1), \ldots, x_n : C^{k_n}(q_n) \vdash^s x_0 \mid p_2 : \phi$, and then by (cut), $x_1 : C^{k_1}(q_1), \ldots, x_n : C^{k_n}(q_n) \vdash^s p_1 \mid p_2 : \phi$ as desired.

  (ii) By lemma 5.2 and the second level ind. hyp. again we obtain $\Gamma \vdash p_1 : \langle\overline{a}b\rangle C^{k-1}(p_1')$, $\Gamma \vdash p_2 : \langle ab\rangle C^{k-1}(p_2')$, and (by the outer ind. hyp.) $x : C^{k-1}(p_1'), y : C^{k-1}(p_2'), \Gamma \vdash^s x \mid y : \phi'$. Then, by $(sync)$ and (cut), $\Gamma \vdash^s p_1 \mid p_2 : \phi$.

  (iii) The proof is analogous, except that $(sync^{\langle\rangle}_{ex})$ is used in place of $(sync)$.

  For case (2), (4) and (5) the proof is analogous to subcase (i) above. Finally, for case (3), assume that $\alpha = \overline{a}(b)$. Either (again leaving out symmetrical subcases) (i) $[\vec{q}/\vec{x}]p_1 \stackrel{\overline{a}(b)}{\rightarrow} p_1'$ and $\models \nu s.p_1' \mid [\vec{q}/\vec{x}]p_2 : \phi'$, or (ii) $b \in s$, $[\vec{q}/\vec{x}]p_1 \stackrel{\overline{a}b}{\rightarrow} p_1'$ and $\models \nu s \backslash \{b\}.p_1' \mid [\vec{q}/\vec{x}]p_2 : \phi'$. In subcase (i) the proof uses $(comp_1^{\langle\rangle})$ and in subcase (ii) the proof uses $(comp^{\langle\rangle}_{1,ex})$.

- $p = x_i$: We obtain from lemma 5.2 and the assumptions that $\models \nu s.q_i : \langle\alpha\rangle\phi'$. Suppose first $\alpha = \tau$. In this case $C^{k_i}(q_i)$ contains a conjunct $\langle\tau\rangle C^{k_i-1}(q_i')$ where $q_i \stackrel{\tau}{\rightarrow} q_i'$ and $\models \nu s.q_i' : \phi'$. By the outer ind. hyp., $x_1 : C^{k_1}(q_1), \ldots, x_i : C^{k_i-1}(q_i'), \ldots, x_n : C^{k_n}(q_n) \vdash^s x_i : \phi'$ and then by (mon) and $\wedge_L$, $x_1 : C^{k_1}(q_1), \ldots, x_i : C^{k_i}(q_i), \ldots, x_n : C^{k_n}(q_n) \vdash^s x_i : \phi$. The cases for $\alpha = \overline{a}b$ and $\alpha = a(b)$ are similar. Assume that $\alpha = \overline{a}(b)$. There are two subcases:

Either (i) $q_i \stackrel{\overline{a}(b)}{\rightarrow} q_i'$ and $\models \nu s.q_i' : \phi'$, or else (ii) $b \in s$, $q_i \stackrel{\overline{a}b}{\rightarrow} q_i'$, and $\models \nu s\backslash\{b\}.q_i' : \phi'$. Subcase (i) follows the previous case. For subcase (ii) we obtain that $x_1 : C^{k_1}(q_1), \ldots, x_i : C^{k_i - 1}(q_i'), \ldots, x_n : C^{k_n}(q_n) \vdash^{s-b} x_i : \phi'$, and then by (mon_{ex}), $x_1 : C^{k_1}(q_1), \ldots, x_i : C^{k_i}(q_i), \ldots, x_n : C^{k_n}(q_n) \vdash^s x_i : \langle\alpha\rangle\phi'$ as desired. Finally assume that $\alpha = ab$. We have $q_i \stackrel{ab}{\rightarrow} q_i'$ and $\models \nu s.q_i' : \phi'$. If $b \in fn(q_i)$ then $C^{k_i}(q_i)$ contains the conjunct $\langle ab\rangle C^{k_i - 1}(q_i')$, and $\Gamma \vdash^s p : \phi$ follows by the ind. hyp. and (mon). If, on the other hand, $b \notin fn(q_i)$ then $C^{k_i}(q_i)$ contains a conjunct of the form $\langle a(c)\rangle C^{k_i - 1}([c/b]q_i')$. Then we can use the ind. hyp. and (mon) along with (btf) to conclude that $\Gamma \vdash^s p : \phi$, completing the case.

$\phi = [\alpha]\phi$. We proceed by cases on $struct(p)$.

- $p = 0$: $\Gamma \vdash^s p : \phi$ by (0).
- $p = a(b).p'$: If $prj(\alpha) \neq a$ then $\Gamma \vdash^s p : \phi$ by (in^2), and if $prj(\alpha) = a$ the proof follows the pattern of the diamond case.
- The cases for $p = \overline{a}b.p'$, $p = \nu a.p'$, $p = p_1 + p_2$, $p = x_i$ follow as in the diamond case.
- $p = p_1 \mid p_2$: We consider only the case for $\alpha = \tau$. The other cases are simpler and left as exercises. Let $p_j' = [q_1/x_1, \ldots, q_n/x_n]p_j$, $j \in \{1,2\}$. Since $\models \nu s.(p_1' \mid p_2') : [\tau]\phi'$ whenever $p_1' \mid p_2' \stackrel{\tau}{\rightarrow} p''$ then $\models p'' : \phi'$ and p'' has the shape $(\nu b.)p_1'' \mid p_2''$ where the νb is optional. By the outermost ind. hyp. we find for each such p'' formulas $\phi_1'(p'')$ and $\phi_2'(p'')$ such that $y_1 : \phi_1'(p''), y_2 : \phi_2'(p'') \vdash^s (\nu b.)y_1 \mid y_2 : \phi'$. Define now

$$
\begin{aligned}
\psi_1 &= \bigwedge\{\phi_1'(p_1' \mid p_2'') \mid p_2' \stackrel{\tau}{\rightarrow} p_2''\} \\
\gamma_1 &= [pact(p_1')^c]\bot \\
\psi_{1,1} &= \bigvee\{\phi_1'(p_1'' \mid p_2) \mid p_1' \stackrel{\tau}{\rightarrow} p_1''\} \\
\psi_{1,2}^{\overline{a}b} &= \bigvee\{\bigwedge\{\phi_1'(p_1'' \mid p_2'') \mid p_2' \stackrel{ab}{\rightarrow} p_2''\} \mid p_1' \stackrel{\overline{a}b}{\rightarrow} p_1''\} \\
\psi_{1,3}^{\overline{a}b} &= \bigvee\{\bigwedge\{\phi_1'(p_1'' \mid p_2'') \mid p_2' \stackrel{\overline{a}b}{\rightarrow} p_2''\} \mid p_1' \stackrel{ab}{\rightarrow} p_1''\} \\
\psi_{1,4}^{\overline{a}}(b) &= \bigvee\{\bigwedge\{\phi_1'(\nu b.p_1'' \mid p_2'') \mid p_2' \stackrel{a(b)}{\rightarrow} p_2''\} \mid p_1' \stackrel{\overline{a}(b)}{\rightarrow} p_1''\} \\
\psi_{1,5}^{\overline{a}}(b) &= \bigvee\{\bigwedge\{\phi_1'(\nu b.p_1'' \mid p_2'') \mid p_2' \stackrel{\overline{a}(b)}{\rightarrow} p_2''\} \mid p_1' \stackrel{a(b)}{\rightarrow} p_1''\}
\end{aligned}
$$

and define symmetrically ψ_2, γ_2, etc. Define then ψ_j', $j \in \{1,2\}$ as in rule $(sync^{\square})$. We need to show $y_1 : \psi_1', y_2 : \psi_2' \vdash^s y_1 \mid y_2 : [\tau]\psi'$. Since we have constructed ψ_j' in the right shape, $(sync^{\square})$ is immediately applicable. Notice then that $\Gamma \vdash^{\emptyset} p_j : \psi$, for all conjuncts ψ of ψ_j', $j \in \{1,2\}$. Consequently, using (cut), $\Gamma \vdash^s p : \psi$, and the case is complete. $\square$

6 Conclusion

This paper is a step towards the definition of automatic or interactive tools for the verification of properties of systems formalized in the π-calculus and described by means of a modal μ-calculus. We intend to pursue this work in two directions:

- To study efficient implementation strategies for the model checker. In particular, it remains to be seen if it is possible to generate the labelled transition system of the process independently from the formula. Recent work on the problem of checking bisimilarity might be of interest here [MP95].
- To extend the compositional proof system to recursive properties by adding rules of discharge in the style of the model checker termination conditions. Such an extension was presented in [Dam95] for CCS, and in [Dam96] this work was extended to the π-calculus and a modal logic which is related to the logic considered in the present paper.

References

[AD95] R. Amadio and M. Dam. Reasoning about higher-order processes. In *Proc. CAAP 95, Aarhus*. SLNCS 915, 1995. Preliminary version appeared as SICS RR 94-18, available at `http://wwwi3s.unice.fr/~amadio/`.

[ALT95] R. Amadio, L. Leth, and B. Thomsen. From a concurrent λ-calculus to the π-calculus. In *Proc. Foundations of Computation Theory 95, Dresden*. SLNCS 965, 1995. Expanded version appeared as ECRC-TR-95-18, available at `http://wwwi3s.unice.fr/~amadio/`.

[AP94] R. Amadio and S. Prasad. Localities and Failures (Extended Summary). In P S Thiagarajan, editor, *Proceedings of* 14^{th} *FST and TCS Conference, FST-TCS'94*, volume 880 of *SLNCS*, pages 205–216. Springer-Verlag, 1994. Preliminary version appeared as ECRC-TR-94-18, Munich, available at `http://wwwi3s.unice.fr/~amadio/`.

[ASW94] H. Andersen, C. Stirling, and G. Winskel. A compositional proof system for the modal μ-calculus. In Proc. LICS'94, 1994.

[Dam93] 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.

[Dam95] M. Dam. Compositional proof systems for model checking infinite state processes. In *Proc. CONCUR'95,* Lecture Notes in Computer Science, 962:12–26, 1995.

[Dam96] M. Dam. On the verification of mobile process networks. Submitted for publication, 1996.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, **32**:137–162, 1985.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, **27**:333–354, 1983.

[Mil92] R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.

[MP95] U. Montanari and M. Pistore. Checking bisimilarity for finitary π-calculus. In *CONCUR, SLNCS*, 1995.

[MPW92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Process, Parts 1-2. *Information and Computation*, 100(1):1–77, 1992.

[MPW93] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114:149–171, 1993.

[OP92] F. Orava and J. Parrow. An algebraic verification of a mobile network. *Formal Aspects of Computing*, pages 497–543, 1992.

[Ora94] F. Orava. *On the Formal Analysis of Telecommunication Protocols*. PhD thesis, Dept. of Computer Systems, Uppsala University and Swedish Institute of Computer Science, 1994.

[San92] D. Sangiorgi. *Expressing mobility in process algebras: first-order and higher order paradigms*. PhD thesis, University of Edinburgh, September 1992.

[San94] D. Sangiorgi. Locality and non-interleaving semantics in calculi for mobile processes. In *Proceedings of TACS 94*. Springer-Verlag, 1994.

[Sti87] C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–347, 1987.

[SW91] C. Stirling and D. Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89:161–177, 1991.

[VD80] D. Van Daalen. *The language theory of Automath*. PhD thesis, Technological University of Eindhoven, 1980. PhD thesis.

[Wal94] D. Walker. Objects in the π-calculus. *Information and Computation*, 1994. (To appear).

[Win89] G. Winskel. Model checking the modal ν-calculus. *Springer Lecture Notes in Computer Science*, 372, 1989.