Proof Systems for π-Calculus Logics

Mads Dam*
Dept. of Teleinformatics, Royal Institute of Technology, Sweden

Abstract

In this paper we study the problem of verifying general temporal and functional properties of mobile and dynamic process networks, cast in terms of the pi-calculus. Much of the expressive power of this calculus derives from the combination of name generation and communication (to handle mobility) with dynamic process creation. In the paper we introduce the π-μ-calculus, an extension of the modal mu-calculus with name equality, inequality, first-order universal and existential quantification, and primitives for name input and output as an appropriate temporal logic for the pi-calculus. A compositional proof system is given with the scope of verifying dynamic networks of pi-calculus agents against properties specified in this logic. The proof system consists of a local part based, roughly, on the classical sequent calculus extended with data structures for private names, and rules to support process structure dependent reasoning. In addition the proof system contains a rule of discharge to close well-founded cycles in the proof graph. The proof system is shown to be sound in general and weakly complete for the non-recursive fragment of the specification logic. We also obtain a weak completeness result for recursive formulas against finite-control calculus processes. Two examples are considered. The first example is based on Milner's encoding of data types into the pi-calculus, specifically the natural numbers. This encoding is interesting from the point of view of verification, since it makes essential use of all the distinguishing features of the pi-calculus, including dynamic process creation. Corresponding to the encoding of natural numbers into the π-calculus we propose an encoding of the type of natural numbers into the π-μ-calculus and establish some type correctness properties. As the second example we consider a garbage-collecting unbounded buffer (which dynamically create and destroy buffer cells) and show how to establish absence of spurious output of such a system.

1 Introduction

In this paper we study the problem of verifying general temporal and functional properties of mobile and dynamic process networks. In such a network processes can be created, and process interconnection topology can be modified during

*Work partially supported by a Swedish Foundation for Strategic Research Junior Individual Grant. Work was partly done at Swedish Institute of Computer Science. Adress: SICS, Box 1263, S-164 28 Kista, Sweden. Email: mfd@sics.se.

execution. Mobility is often achieved by a mechanism for generating inter-process links, and passing them between processes. For instance, in the π-calculus [12] links are primitive names of communication channels, and in the programming language Erlang [3] links serve as both communication channels and process identifiers. The combination of mobility with dynamic creation of processes is very powerful. For the case of π-calculus this is witnessed by the encodings into the π-calculus of, e.g., data types, functions, objects, and higher-order processes [10, 11, 19, 16]. But it is also a power which is used extensively in everyday programming practice, to dynamically set up data and process structures, to adapt applications and systems to change in their environments, to support fault tolerance and code replacement in running systems, to name just a few scenarios (cf. [3]).

The cost of this power is the unbounded and essential growth of state spaces as processes compute, rendering analyses by global state space exploration in general impossible. An alternative which was explored in [5] for CCS is to take a compositional, proof-based approach. In this paper we extend this approach to the π-calculus and show how a compositional proof system can be built with the scope of verifying quite general properties of π-calculus processes.

It is important to note that this is an ambitious and difficult task. There are few approaches to verification around that can deal satisfactorily with this kind of problem even in settings that are computationally simpler than that of the π-calculus. We do not claim to give the definite answer to this problem in this paper — much more work will be required before stable methods and criteria for measuring their usefulness have been found. What we do claim, however, is to present *one* possible approach that does, according to some set of criteria (soundness, weak forms of completeness, non-trivial examples) address and adequately solve the problem.

The investigation is cast in terms of judgments of the form $\Gamma \vdash^s E : \phi$ where E is an open π-calculus term, ϕ is a general temporal formula, Γ is a sequence of assumptions governing agent variables free in E, and s is a set of channel names local to E. Verification in such a setting must be inherently compositional, since the behaviour of E is defined only up to properties of its constituent parts, as determined by Γ. Temporal specifications are given in an extension of the modal μ-calculus with name equality, inequality, first-order universal and existential quantification, and primitives for name input and output along the lines of [4]. We present a proof system for judgments which is sound and weakly complete for recursion-free formulas, and for general formulas we show that the proof system is sound in general and weakly complete for finite control processes. As the CCS-based proof system of [5] the proof system consists of a (rather large) number of proof rules that account, roughly, for the modal fragment, plus a single rule of discharge to handle fixed points.

We illustrate the use and scope of the proof system by means of two examples. First an example based on Milner's encoding of data types into the π-calculus shows how the type of natural numbers can be encoded. While we have no inherent interest in the natural numbers the example is of interest since it exercises all important aspects of both the π-calculus (dynamic process creation, name generation, scope extrusion, and communication), and of the

temporal logic (fixed points, alternation, quantification, equality and inequality, input and output). Moreover, it illustrates well an important point of the proof system, namely its capacity to uniformly prove properties pertaining to infinite collections of essentially distinct agents. Finally the example is surprisingly subtle: It is not at all trivial to arrive at a suitable encoding of the property of "representing a natural number", let alone to formally prove type correctness properties such as those we consider.

As a second example we consider bounded and (in particular) unbounded buffers and show, as the main example, absence of spurious output from an unbounded directional buffer which is "garbage-collecting" in the sense that buffer elements which become empty are terminated.

The chosen setting for the π-calculus is introduced in section 2. We work with a largely standard version of the polyadic π-calculus [10], using recursive process definitions and incorporating a standard conditional. We then proceed to introduce the "π-μ-calculus", a version of first-order μ-calculus in the style of Park [14], extended with π-calculus-specific primitives. In section 4 we present a few example specifications, notably some examples of specifications addressing basic buffer properties (order preservation, absence of spurious output, absence of message loss) and a formalisation of the "type of natural numbers". Then we proceed to present the modal fragment of the proof system, the shape of judgments, their semantics, and the proof rules along with a few example derivations. The modal fragment of the proof system is grouped naturally into several collections of rules. Some (the logical rules, governing the first-order connectives) are largely standard. Others (the structural rules and — to a slightly less extent — the rules for equality and inequality) are somewhat interesting in the way private names are handled. Two further collections of rules are the rules for transition modalities and the rules for modalities reflecting name input and output, the io-modalities. The modal fragment of the proof system is shown to be sound and weakly complete in section 6. We then proceed to consider fixed points. In section 7 the semantics and proof rules for fixed point formulas is introduced. The handling of fixed points follows a novel approach, introduced first in [6]. This approach exploits approximation ordinals in an explicit way paving the way for a logical account which is far simpler and more elegant than the story given in [5]. The proof rules are grouped in two: First is a collection of local rules, providing support for fixed point unfolding, and introduction and unfolding of approximated formulas. The second and final part of the proof system consists of a single rule of discharge, providing, roughly, a formal correlate of well-founded induction. In section 8, then, we go on to prove a weak form of completeness, by reducing proofs in a "global", model-checking oriented version of the proof system which is known to be complete, to proofs in the compositional system. Section 9 and 10, then contains verification examples pertaining to the natural number and buffer examples and section 11, finally, contains our conclusions and pointers for future work. Proofs of the main completeness results have been deferred to appendices.

2 Preliminaries on the pi-calculus

In this section we introduce the π-calculus, its syntax and operational semantics.

Syntax Assume denumerable sets of *agent (abstraction) identifiers* D and of (channel) *names* $a, b, c \in \mathcal{G}$. We use $\tilde{a}$ to denote vectors $a_1, \ldots, a_n$. *Processes* $P, Q \in \mathcal{P}$ along with *abstractions*, $A \in \mathcal{A}$, and *concretions*, $C \in \mathcal{C}$, are generated by the following abstract syntax:

$$\begin{array}{rcl} P & ::= & 0 \mid P + P \mid \tau.P \mid a.A \mid \overline{a}.C \mid P \mid P \mid \textbf{if } a = b \textbf{ then } P \textbf{ else } P \mid \\ & & \nu a.P \mid D(\tilde{a}) \\ A & ::= & (a)A \mid P \\ C & ::= & \langle a \rangle C \mid \nu a.C \mid P \end{array}$$

The notation is largely standard. We use a standard if-then-else notation for the conditional in place of the matching/mismatching notations $[a = b]P$ and $[a \neq b]P$ more common in π-calculus contexts. This is to avoid excessive proliferation of the "box" notation which is used later also for one of the modal operators.

Agents are interpreted relative to an environment which determines, for each agent identifier D, a defining equation $D(\tilde{a}) \triangleq P$. An alternative to such recursive definitions is to use the "bang" operator $!P$, easily definable by the identity $!P = P \mid !P$. It is also possible, though a little more involved, to derive recursively defined processes using the "bang".

Binding The calculus has two binding operators:

- Abstractions $(a)A$ binds a in A
- Restrictions $\nu a.P$ ($\nu a.C$) binds a in P (in C).

Agent expressions are considered only up to renaming of bound names. $fn(P)$, $bn(P)$, and $n(P)$ are the free names, bound names, and names (free or bound) of P, respectively. We use $P\{a_1/b_1, \ldots, a_n/b_n\}$, or in vectorized form, $P\{\tilde{a}/\tilde{b}\}$, as the notation for uniform and simultaneous substitution, here of names.

Actions An *action*, α, is either the internal action τ, an *input action* $a(\tilde{b})$, or an *output action* of the shape $\nu\tilde{b}.\overline{a}\langle\tilde{c}\rangle$ where $\tilde{b} \subseteq \tilde{c}$ and $a \notin \tilde{b}$. Actions are used as derived notation, by the clauses

$$\begin{array}{rcl} a(b_1, \ldots, b_n).P & \triangleq & a.(b_1) \cdots (b_n)P \\ \nu b_1, \ldots, b_n.\overline{a}\langle c_1, \ldots, c_m\rangle.P & \triangleq & \nu b_1.\cdots.\nu b_n.\overline{a}.\langle c_1\rangle \cdots \langle c_m\rangle P \end{array}$$

By means of the identifications

$$\begin{array}{rcll} \nu a.\nu b.C & = & \nu b.\nu a.C & \\ \nu a.\langle b\rangle C & = & \langle b\rangle \nu a.C & (\text{if } a \neq b) \\ \overline{a}.\nu b.C & = & \nu b.\overline{a}.C & (\text{if } a \neq b) \end{array}$$

it is possible to rewrite each process $a.A$ or $\overline{a}.C$ into one of the shape $\alpha.P$. Given the binding conventions, the functions fn, bn and n are extended to actions in the obvious way.

Transition Semantics The intended semantics is quite well known from CCS [9] and the original π-calculus papers [12]:

- 0 is the inert process, incapable of performing transitions.
- $P+Q$ is the CCS choice construction: A transition of $P+Q$ is a transition of either P or of Q.
- $\alpha.P$ is an agent offering just one transition, labelled with α, with P as the resulting next state.
- Actions of the shape τ represent internal, or unobservable, actions. Actions of the shape $a(\tilde{b})$ represents input actions, and actions of the shape $\nu\tilde{b}.\overline{a}\langle\tilde{c}\rangle$ represents output actions.
- In the case $\alpha = \nu b.\overline{a}.\langle b\rangle$ the process concerned is emitting a private name b to the receiver, causing the scope of that b to be *extruded*, i.e. extended, possibly involving alpha conversions, to encompass the receiving process.
- $P \mid Q$ is parallel composition offering the transitions of P and Q separately, as well as internal transitions arising from (synchronous) communication between P and Q. Communications cause input and output actions to be matched.
- **if** $v_1 = v_2$ **then** P_1 **else** P_2 is the conditional.
- $\nu a.P$ declares a new name a to be used by P.
- $D(\tilde{a})$ is invocation of the process defined by D.

Example 2.1 The agent D defined by

$$D(a) = (\nu b).\overline{a}\langle b\rangle.b(c).D(c)$$

is a recursive agent, parametrised on a, which declares a new local name b which is passed along a to the environment whereupon a name c can be received along b, resulting in the agent $D(c)$.

The transition rules are given in Table 1. The table does not include symmetric versions of the rules Sum, Com, and Par. Those should be assumed. Since terms are identified up to alpha-conversion many side conditions intending to prevent confusion of scope can be avoided.

Example 2.2 Let $P = a(b).P_1$ and $Q = \nu c.\overline{a}\langle c\rangle Q_1$. Up to choice of bound names there are three transitions enabled from $P \mid Q$, namely

- $P \mid Q \stackrel{a(d)}{\rightarrow} (P_1\{d/b\} \mid Q)$,

$$\textsc{Sum}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \textsc{Pre}\ \frac{\cdot}{\alpha.P \xrightarrow{\alpha} P}$$

$$\textsc{Com}\ \frac{P \xrightarrow{\nu\tilde{b}.\overline{a}\langle\tilde{b'}\rangle} P' \quad Q \xrightarrow{a(\tilde{c})} Q'}{P \mid Q \xrightarrow{\tau} \nu b.(P' \mid (Q'\{\tilde{b'}/\tilde{c}\}))} \qquad \textsc{Par}\ \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

$$\textsc{If}_1\ \frac{P \xrightarrow{\alpha} P'}{\textbf{if } a = a \textbf{ then } P \textbf{ else } Q \xrightarrow{\alpha} P'}$$

$$\textsc{If}_2\ \frac{Q \xrightarrow{\alpha} Q'}{\textbf{if } a = b \textbf{ then } P \textbf{ else } Q \xrightarrow{\alpha} Q'} \quad (a \neq b)$$

$$\textsc{Res}\ \frac{P \xrightarrow{\alpha} P'}{\nu a.P \xrightarrow{\alpha} \nu a.P'} \quad (a \notin n(\alpha))$$

$$\textsc{Open}\ \frac{P \xrightarrow{\nu\tilde{b}.\overline{a}\langle\tilde{b'}\rangle} P'}{\nu c.P \xrightarrow{\nu\tilde{b}.c.\overline{a}\langle\tilde{b'}\rangle} P'} \quad (c \neq a, c \in \tilde{b'} \backslash \tilde{b})$$

$$\textsc{Id}\ \frac{P\{\tilde{b}/\tilde{a}\} \xrightarrow{\alpha} P'}{D(\tilde{b}) \xrightarrow{\alpha} P'} \quad (D(\tilde{a}) \triangleq P)$$

Table 1: Transition rules

- $P \mid Q \xrightarrow{\nu d.\overline{a}\langle d\rangle} (P \mid Q_1\{d/c\})$, and
- $P \mid Q \xrightarrow{\tau} \nu d.(P_1\{d/b\} \mid Q_1\{d/c\})$

We conclude the section by introducing the two running examples of the paper.

Example 2.3 (Buffers) A 1-place buffer reads a name from an input port i and delivers it to an output port o:

$$Buf_1(i, o) = i(a).\overline{o}\langle a\rangle.Buf_1(i, o). \tag{1}$$

Inductively, if Buf_{n_1} (Buf_{n_2}) is an n_1-place (n_2-place) buffer abstraction similar to Buf_1 then

$$Buf_{n_1+n_2}(i, o) = \nu m.(Buf_{n_1}(i, m) \mid Buf_{n_2}(m, o)) \tag{2}$$

is an $n_1 + n_2$-place buffer abstraction. Unbounded buffers need to be able to allocate new buffer cells dynamically:

$$Buf(i, o) = i(a).\nu m.(Buf(i, m) \mid \overline{o}\langle a\rangle.Buf(m, o)). \tag{3}$$

Whenever a data item is input by Buf a new buffer cell is allocated, never to be deallocated. This is clearly wasteful. We may also consider an unbounded buffer with reclaimable cells, organised as a linked list as shown in fig. 1. The main component is the buffer cell $BufCell(o, d, n_l, n_r)$ which holds the value d and waits to either output d along o and then terminate and communicate o and n_r "upstream" along n_l, or else receive a new output port o' and a new "down-link" n' along n:

$$BufCell(o, d, n_l, n_r) \quad = \quad \overline{o}\langle d\rangle.\overline{n_l}\langle o, n_r\rangle.0 + n_r(o', n).BufCell(o', d, n_l, n) \tag{4}$$

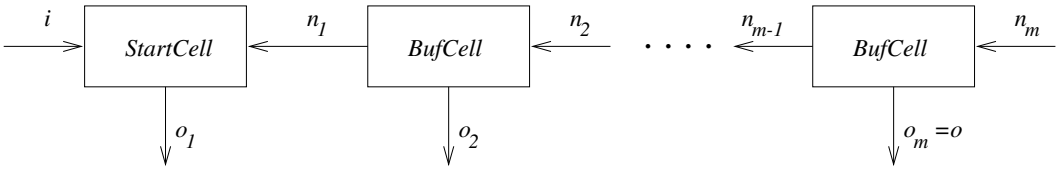

Figure 1: Buffer with reclaimable cells

Start cells are responsible for the creation of buffer cells:

$$\begin{aligned} StartCell(i,o,n) &= i(d).\nu o'.\nu n'.(StartCell(i,o',n') \mid BufCell(o,d,n',n)) + \\ &\quad n(o',n').StartCell(i,o',n') \end{aligned}$$

A "garbage collecting" unbounded buffer then consists initially of just a start cell:

$$GCBuf(i,o) = \nu n.StartCell(i,o,n) \tag{5}$$

Example 2.4 (Natural Numbers) We consider an encoding of natural numbers based on Milner's encoding of data types in the polyadic π-calculus (cf. [10]). The idea is to represent a natural number "located" at a name n as a process which expects two names, say s and z, along n, and then proceed to either synchronize on z to signal to the receiver that the "value" of n is 0, or else to output the location n' of another natural number along s to signal that the value of n is $n'+1$.

Define the constants $ZERO$ and $SUCC$ in the following way:

$$\begin{aligned} ZERO(n) &= n(s,z).\overline{z}\langle\rangle.0 \\ SUCC(n,m) &= m(s,z).\overline{s}\langle n\rangle.0. \end{aligned}$$

Given a location n, $ZERO(n)$ inputs two names along n to attempt an output along the second of the two, the "zero" channel. Similarly, $SUCC(n,m)$ receives two names along m to attempt an output along the first, the "successor" channel, passing along it the location of n. Thus we can represent, eg., the natural number 1 as the agent

$$ONE(one) = \nu zero.(SUCC(zero,one) \mid ZERO(zero)).$$

A variety of operations on naturals can be defined, including addition, multiplication, and as a very basic one the "copying" operation

$$\begin{aligned} COPY(n,m) &= \nu s_1.\nu z_1.\overline{n}\langle s_1,z_1\rangle. \\ &\quad z_1().ZERO(m) + \\ &\quad s_1(n_1).\nu m_1.(SUCC(m_1,m) \mid COPY(n_1,m_1)) \end{aligned}$$

that turns a natural at location n into a natural at location m. Addition can be defined as follows:

$$\begin{aligned} ADD(n_1,n_2,m) &= \nu s_1.\nu z_1.\overline{n_1}\langle s_1,z_1\rangle.(z_1().COPY(n_2,m)) + \\ &\quad (s_1(n_{11}).\nu m_1.(SUCC(m_1,m) \mid ADD(n_{11},n_2,m_1))) \end{aligned}$$

3 A π-μ-Calculus

In this section we introduce the logic which we use for specifying properties of agents. The logic is a first-order version of the modal μ-calculus extended with operations that describe input/output behaviour.

Syntax Assume a denumerable set of *predicate variables* X. *Formulas* $\phi, \psi \in \mathcal{F}$ in the π-μ-calculus are generated in the following way:

$$\begin{aligned}\phi \quad ::= \quad & a = b \mid a \neq b \mid \forall a.\phi \mid \exists a.\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid <l_\tau>\phi \mid [l_\tau]\phi \\ & \mid a \rightarrow \phi \mid a \leftarrow \phi \mid \nu a \rightarrow \ \phi \mid \nu a \leftarrow \phi \mid X(\vec{a}) \mid (\nu X(\vec{a}).\phi)(\vec{b}) \mid (\mu X(\vec{a}).\phi)(\vec{b})\end{aligned}$$

In this grammar we use l to range over names, a, and co-names, $\overline{a}$, and we use l_τ to range over names, co-names, and τ. A *monadic* formula is one with strict alternation between "transition" modalities (of the shape $<l>\phi$ or $[l]\phi$) and "io" modalities (of the shape $a \rightarrow \phi$, $a \leftarrow \phi$, $\nu a \rightarrow \phi$, or $\nu a \leftarrow \phi$), where we also require that any outermost modal operator be a transition modality.

The Connectives The language is based on a first-order modal μ-calculus with name equality and inequality, universal and existential name quantification, boolean connectives "and" ($\wedge$) and "or" ($\vee$), modal operators $<l>$ and $[l]$, and (parametrised) least (μ) and greatest (ν) fixed points. A *modal* formula is a formula in the fragment with neither fixed points nor predicate variables. An *elementary* formula is a formula in the first-order language of equality (over names). It is often convenient to treat elementary formulas differently from other formulas as they do not depend on the agent being predicated.

The connectives $a \rightarrow$, $a \leftarrow$, $\nu a \rightarrow$, and $\nu a \leftarrow$ are used for input, free output, input of a fresh name, and bound output, respectively. For instance, $a \rightarrow \phi$ predicates an abstraction and is taken to mean that, when applied to the name a, the resulting agent satisfies ϕ. Observe that a is not bound in $a \rightarrow \phi$. Similarly $a \leftarrow \phi$ predicates a concretion term, and states that the term is of the shape $\langle b \rangle \phi$ such that a and b are equal and ϕ holds of the continuation. For bound outputs the operation $\nu a \leftarrow \phi$ is used. The operation $\nu a \rightarrow \phi$ was introduced in a slightly different shape in [1]. This connective predicates abstractions and means that whenever the agent being predicated inputs a name a which is fresh then ϕ holds of the continuation. Both the modalities $\nu a \rightarrow \phi$ and $\nu a \leftarrow \phi$ bind a in ϕ. The syntax presented here divorces handling of transition labels, or subjects, from handling of the parameters, or objects. An alternative which we return to in more detail below, is to devise connectives in the style of those used by Milner et al [13] which combine the two, in modalities which reflect action capabilities more directly.

Example 3.1 The formula

$$(\nu X(a).<\overline{a}>\nu b \leftarrow [b]\forall c.c \rightarrow X(c))(a)$$

expresses of an agent that it can perform a bound output of a b along channel a. After this whenever a c is received along b the continuation satisfies $X(c)$.

Bindings, Sentences We use σ to range over $\{\nu, \mu\}$, and (l) to range over the modalities $<l>$ and $[l]$. Formulas are considered only up to alpha-renaming of bound predicate and name variables. Name binders are the first-order quantifiers, the bound output connective, and the fixed point operators. For instance, in $(\nu X(\tilde{a}).\phi)(\tilde{b})$, names in $\tilde{b}$ occur freely and names in $\tilde{a}$ bind their occurrences in ϕ. We can without loss of generality assume that fixed point formulas do not contain free names. A *sentence* is a formula that do not contain free occurrences of predicate variables. Unless otherwise specified we restrict attention to sentences.

Predicate and Name Interpretations Names serve a double role, both as constants (two distinct names occurring freely at the top level of a process term are regarded as distinct) and as variables (since names can be bound and instantiated). A *name interpretation* is a mapping $\eta : \mathcal{G} \to \mathcal{G}$ which assigns names (as values) to names (as variables). We require of name interpretations η that $\mathcal{G} - \mathit{range}(\eta)$ is an infinite set. A name $b \in \mathcal{G}$ is said to be *fresh for* η if $b \notin \mathit{range}(\eta)$, and $\eta\{b/a\}$ is the update of η that maps a to b and otherwise acts as η. Let $\nu b.\eta = \eta\{b/b\}$, and if $s = \{a_1, \ldots, a_n\}$ then $\nu s.\eta = \nu a_1.\cdots.\nu a_n.\eta$. Observe that the operation $\nu b.-$ is generally only applied to η for which b is fresh. Name interpretations are extended to actions by the clause $\eta(\tau) = \tau$.

Predicate variables depend for their semantics upon a list of argument names and a name interpretation. Thus, a predicate variable interpretation can be taken to be a mapping $\rho(X)(\eta, \tilde{a}) \subseteq \mathcal{P}$. Maps $f : (\eta, \tilde{a}) \mapsto S \subseteq \mathcal{P}$ are ordered by $\leq$ defined as subset containment lifted pointwise to functions.

Semantics, Validity The semantics is given in terms of a mapping $\|\phi\|\rho\eta \subseteq \mathcal{P}$ where ρ is a predicate variable interpretation and η is a name interpretation. The definition of $\|\cdot\|$ is given in table 2. In the clauses for bound input and bound output the notation *c fresh* means that c does not occur freely in neither A nor ϕ, and c is fresh for η. For sentences ϕ the ρ is immaterial, and we abbreviate $A \in \|\phi\|\rho\eta$ by $A \in \|\phi\|\eta$. Sometimes we write $\models_\eta A : \phi$ in place of $A \in \|\phi\|\eta$. A sentence ϕ is said to be *valid* if $\|\phi\|\eta = \mathcal{P}$ for all name interpretations η. Sentences ϕ and ψ are *equivalent*, written $\phi \equiv \psi$, if for all η, $\|\phi\|\eta = \|\psi\|\eta$.

The clauses for the modal operators lead to derived notation for the transition relation:

$$P \xrightarrow{a} (\vec{b})Q \quad \text{iff} \quad P \xrightarrow{a(\vec{b})} Q$$
$$P \xrightarrow{\overline{a}} \nu\vec{b_1}.\langle\vec{b_2}\rangle Q \quad \text{iff} \quad P \xrightarrow{\nu\vec{b_1}.\overline{a}.\langle\vec{b_2}\rangle} Q$$

We can thus simplify the transition modality clauses in the following style:

$$\|<a>\phi\|\rho\eta = \{P \in \mathcal{P} \mid \exists c, A.P \xrightarrow{c} A, c = \eta(a), \text{ and } A \in \|\phi\|\rho\eta\}.$$

Abbreviations We introduce the following derived forms:

$$\top \triangleq \forall a.a = a \qquad \bot \triangleq \forall a.a \neq a$$

$$
\begin{array}{rcl}
\|a = b\|\rho\eta & = & \{A \in \mathcal{P} \cup \mathcal{A} \cup \mathcal{C} \mid \eta(a) = \eta(b)\} \\
\|a \neq b\|\rho\eta & = & \{A \in \mathcal{P} \cup \mathcal{A} \cup \mathcal{C} \mid \eta(a) \neq \eta(b)\} \\
\|\forall a.\phi\|\rho\eta & = & \bigcap\{\|\phi\|\rho\eta\{b/a\} \mid b \in \mathcal{G}\} \\
\|\exists a.\phi\|\rho\eta & = & \bigcup\{\|\phi\|\rho\eta\{b/a\} \mid b \in \mathcal{G}\} \\
\|\phi \wedge \psi\|\rho\eta & = & \|\phi\|\rho\eta \cap \|\psi\|\rho\eta \\
\|\phi \vee \psi\|\rho\eta & = & \|\phi\|\rho\eta \cup \|\psi\|\rho\eta \\
\|<\tau>\phi\|\rho\eta & = & \{P \in \mathcal{P} \mid \exists Q.P \xrightarrow{\tau} Q \text{ and } Q \in \|\phi\|\rho\eta\} \\
\|<a>\phi\|\rho\eta & = & \{P \in \mathcal{P} \mid \exists c, \vec{b}, Q, P \xrightarrow{c(\vec{b})} Q, c = \eta(a), \text{and } (\tilde{b})Q \in \|\phi\|\rho\eta\} \\
\|<\overline{a}>\phi\|\rho\eta & = & \{P \in \mathcal{P} \mid \exists \vec{b_1}, c, \vec{b_2}, Q.P \xrightarrow{\nu\vec{b_1}.c\langle\vec{b_2}\rangle} Q, c = \eta(a), \\
 & & \qquad \text{and } \nu\vec{b_1}.\langle\vec{b_2}\rangle Q \in \|\phi\|\rho\eta\} \\
\|[\tau]\phi\|\rho\eta & = & \{P \in \mathcal{P} \mid \forall Q.P \xrightarrow{\tau} Q \text{ implies } Q \in \|\phi\|\rho\eta\} \\
\|[a]\phi\|\rho\eta & = & \{P \in \mathcal{P} \mid \forall c, \vec{b}, Q, P \xrightarrow{c(\vec{b})} Q, c = \eta(a) \text{ implies } (\tilde{b})Q \in \|\phi\|\rho\eta\} \\
\|[\overline{a}]\phi\|\rho\eta & = & \{P \in \mathcal{P} \mid \forall \vec{b_1}, c, \vec{b_2}, Q.P \xrightarrow{\nu\vec{b_1}.c\langle\vec{b_2}\rangle} Q, \\
 & & \qquad c = \eta(a) \text{ implies } \nu\vec{b_1}.\langle\vec{b_2}\rangle Q \in \|\phi\|\rho\eta\} \\
\|a \rightarrow \phi\|\rho\eta & = & \{(b)A \mid A\{\eta(a)/b\} \in \|\phi\|\rho\eta\} \\
\|a \leftarrow \phi\|\rho\eta & = & \{\langle b\rangle C \mid \eta(a) = b \text{ and } \ C \in \|\phi\|\rho\eta\} \\
\|\nu a \rightarrow \phi\|\rho\eta & = & \{(b)A \mid \forall c.c \text{ fresh implies } A\{c/b\} \in \|\phi\|\rho(\nu c.\eta)\} \\
\|\nu a \leftarrow \phi\|\rho\eta & = & \{\nu b.\langle b\rangle C \mid \exists c.c \text{ fresh and } \ C\{c/b\} \in \|\phi\|\rho\eta\{c/a\}\} \\
\|X(\tilde{a})\|\rho\eta & = & \rho(X)(\eta, \tilde{a}) \\
\|(\nu X(\tilde{a}).\phi)(\tilde{b})\|\rho\eta & = & (\sqcup\{f \mid f \leq M(f)\})(\eta, \tilde{b}), \ (M(f)(\eta, \tilde{c}) = \|\phi\|\rho\{f/X\}\eta\{\tilde{c}/\tilde{a}\}) \\
\|(\mu X(\tilde{a}).\phi)(\tilde{b})\|\rho\eta & = & (\sqcap\{f \mid M(f) \leq f\})(\eta, \tilde{b}), \ (M \text{ as above})
\end{array}
$$

Table 2: Formula semantics

$$isP \triangleq [\tau]\top$$

$$isA \triangleq \forall a.a \rightarrow \top \qquad isCf \triangleq \exists a.a \leftarrow \top \qquad isCb \triangleq \nu a \leftarrow \top$$

Observe that isP holds of processes, isA of (non-trivial) abstractions, $isCf$ of free outputs and $isCb$ of bound outputs. Let $\mathcal{S}$ range over these four "typing formulas". Two typing formulas are said to be *complementary* if one of them is isA and the other either $isCf$ or $isCb$.

An alternative to the definitions of $\top$ and $\bot$ is to set $\top \triangleq (\nu X().X())()$ and similarly $\bot \triangleq (\mu X().X())()$. These definitions are easily shown to be equivalent to the original ones in the proof system below.

The logic is closed under negation, as we can define the operation $\neg$ by the usual De-Morgan rules plus:

$$\begin{aligned}
\neg{<\alpha>}\phi &= ([\alpha]\neg\phi) \vee isA \vee isCf \vee isCb \\
\neg[\alpha]\phi &= (<\alpha>\neg\phi) \vee isA \vee isCf \vee isCb \\
\neg(a \rightarrow \phi) &= (a \rightarrow \neg\phi) \vee isP \vee isCf \vee isCb \\
\neg(a \leftarrow \phi) &= (\exists b.a \neq b \wedge b \leftarrow \top) \vee (a \leftarrow \neg\phi) \vee isP \vee isA \vee isCb \\
\neg(\nu a \rightarrow \phi) &= (\nu a \rightarrow \neg\phi) \vee isP \vee isCf \vee isCb \\
\neg(\nu a \leftarrow \phi) &= (\nu a \leftarrow \neg\phi) \vee isP \vee isA \vee isCf
\end{aligned}$$

With classical negation a number of standard abbreviations like $\phi \supset \psi \triangleq \neg\phi \vee \psi$ becomes available (but observe that recursion variables can in general only be negated an even number of times). There would in principle be no problem in taking negation as primitive. We choose not to do so, mainly as a matter of convenience, as otherwise the proof rules, which are cluttered up enough as they stand, would become even harder to read.

π-Calculus Modalities Comparing with earlier attempts to define modal and temporal logics for the π-calculus we consider first the modal logic of Milner, Parrow, and Walker [13]. Their basic modalities are rendered in our framework as follows:

$$\begin{aligned}
<\overline{a}b>\phi &\triangleq <\overline{a}>b \leftarrow \phi \\
<ab>\phi &\triangleq <a>b \rightarrow \phi \\
<\overline{a}(b)>\phi &\triangleq <\overline{a}>\nu b \leftarrow \phi \\
<a(b)>\phi &\triangleq <a>\exists b.b \rightarrow \phi \\
<a(b)>^{\mathrm{L}}\phi &\triangleq <a>\forall b.b \rightarrow \phi \\
<a(b)>^{\mathrm{E}}\phi &\triangleq \forall b.<a>b \rightarrow \phi
\end{aligned}$$

We refer to these operators as the "MPW modalities". We leave aside the issue of whether a reduction in the other direction exists (of modal monadic formulas to formulas in the logic of [13]). Trivially this is not so since the logic of [13] lacks a modality for bound input. With the addition of such a modality the matter is less easily resolved, however.

Other logics for π-calculus have been proposed in work by Milner [10], Dam [4], and Amadio and Dam [1]. In [10] dependent sum and product constructions were proposed, used also in [4]:

$$\begin{aligned}\Sigma a.\phi &\triangleq (\exists a.a \leftarrow \phi) \vee (\nu a \leftarrow \phi)\\ \forall' a.\phi &\triangleq \forall a.a \rightarrow \phi\\ \exists' a.\phi &\triangleq \exists a.a \rightarrow \phi\end{aligned}$$

We use "primed" versions of the quantifiers here so as not to confuse with the standard first-order quantifiers. Observe that, even in the absence of bound input modalities, the logics of [10, 4] are strictly less expressive than that of [13] as the former does not separate free and bound output. The modalities of [1], finally, are easily derivable as well. We leave out the details.

Logical Characterisation An important property of a modal logic such as the one considered here is its capability to separate models. In [13] it was shown that the MPW modalities could be used to characterise a particular, quite strong, process equivalence called late bisimulation equivalence (cf. [12]).

Definition 3.2 (Late Bisimulation Equivalence) A binary relation $\mathcal{R}$ on $\mathcal{P}$ is a *late simulation* if $P_1 \mathcal{R} P_2$ implies:

1. If $P_1 \stackrel{\alpha}{\rightarrow} Q_1$ and α is not an input action then there is some Q_2 such that $P_2 \stackrel{\alpha}{\rightarrow} Q_2$ and $Q_1 \mathcal{R} Q_2$.

2. If $P_1 \stackrel{a(\vec{b})}{\rightarrow} Q_1$ then for some Q_2, $P_2 \stackrel{a(\vec{b})}{\rightarrow} Q_2$ and for all $\vec{c}$, $Q_1\{\vec{c}/\vec{b}\} \mathcal{R} Q_2\{\vec{c}/\vec{b}\}$.

The relation $\mathcal{R}$ is a *late bisimulation* if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are late simulations. P_1 and P_2 are *late bisimilar*, $P_1 \dot{\sim} P_2$, if $P_1 \mathcal{R} P_2$ for some late bisimulation $\mathcal{R}$.

In the second clause of definition (3.2) the arities of vectors $\vec{b}$ and $\vec{c}$ are required to coincide. Observe that the simplicity of the definition, both for input and output actions, relies heavily on alpha-conversion to avoid accidental capture of bound names.

Other equivalences appear in the π-calculus literature. *Early bisimulation* ([12]) is strictly weaker than late bisimulation. It is obtained by swapping the quantifications over Q_2 and $\vec{c}$ in clause 2 of definition (3.2). *Open bisimulation* [15] is strictly stronger than late bisimulation and requires the bisimulation relation to be closed under substitution. *Ground bisimulation*, finally, is standard (CCS) bisimulation equivalence [9] applied to the π-calculus. Thus ground bisimulation avoids quantification entirely in its defining clauses, and it is strictly weaker than early bisimulation equivalence. For a substantially constrained version of the π-calculus, eliminating the conditional, matching (a conditional of the special form **if** $a = b$ **then** P **else** 0), and continuation under output prefix, it can be shown, however, that all four equivalences coincide [15].

We obtain the following logical characterisation result:

Proposition 3.3 (Logical Characterisation) *Two processes P and Q are late bisimilar if and only if for all sentences ϕ and all name interpretations η, $\models_\eta P : \phi$ implies $\models_\eta Q : \phi$.*

PROOF: In [13] the logical characterisation result is proved for the logic constructed using equations, inequations, conjunction, disjunction, and the MPW modalities. The adaptation of this result to the present setting is easy and left out. It thus suffices to show that all formulas in our language respect bisimulation equivalence in the sense that if $\models_\eta P : \phi$ and $P \dot{\sim} Q$ then $\models_\eta Q : \phi$ as well. To cater for fixed points the assertion need to be generalised: Say a predicate interpretation ρ *respects* $\dot{\sim}$ if whenever $P \in \rho(X)(\eta, \tilde{a})$ and $P \dot{\sim} Q$ then $Q \in \rho(X)(\eta, \tilde{a})$ too. We show by induction in the structure of ϕ that if ρ respects $\dot{\sim}$, $P \in \|\phi\|\rho\eta$, and $P \dot{\sim} Q$ then $Q \in \|\phi\|\rho\eta$. The details of this induction are not difficult and left to the reader. □

By constraining the nesting of transition and io modalities a similar characterisation of early bisimulation can be given. In effect the formation of the late input MPW modality $<a(b)>^{\text{L}}$ needs to be prevented, as shown by [13].

4 Example Specifications

In this section we give some examples of agent properties that can be specified using the above logic.

Example 4.1 (Weak modalities) One can easily derive modalities which are insensitive to the number of initial τ-transitions.

$$\begin{aligned} <<>>\phi &\triangleq \mu X.\phi \vee <\tau>X \\ <<l>>\phi &\triangleq <<>><l>\phi \\ [[]]\phi &\triangleq \nu X.\phi \wedge [\tau]X \\ [[l]]\phi &\triangleq [[]][l]\phi \end{aligned}$$

Example 4.2 (Wildcard input and output) Often one wishes to ignore the identity of names being input or output:

$$\begin{aligned} \_ \rightarrow \phi &= \forall a.a \rightarrow \phi \\ \_ \leftarrow \phi &= (\exists a.a \leftarrow \phi) \vee \nu a \leftarrow \phi \end{aligned}$$

This definition presupposes that a does not appear freely in ϕ.

Example 4.3 (Buffer properties) We consider some properties one might like to impose of a π-calculus buffer taking input along the channel i and producing output to the channel o. The formulas in this example are due to Joachim Parrow (personal communication).

- **Order preservation (of first data item).**

$$
\begin{aligned}
&FirstOut(i,o,d) = \\
&\quad (\nu X(d).[\tau]X(d) \wedge \forall a. \\
&\qquad ([\overline{a}](a = o \wedge d \leftarrow true)) \wedge \\
&\qquad ([a](a = i \wedge \_ \rightarrow X(d))))(d)
\end{aligned}
$$

$$
\begin{aligned}
&OrdPres(i,o) = \\
&\quad \nu X.[\tau]X \wedge \forall c. \\
&\qquad ([\overline{c}](c = o \wedge \_ \leftarrow X)) \wedge \\
&\qquad ([c](c = i \wedge \forall d.d \rightarrow FirstOut(i,o,d)))
\end{aligned}
$$

  The idea is quite simple: $OrdPres(i,o)$ holds until something (the d) is input along i. Then $FirstOut(i,o,d)$ takes effect. This fixed point holds invariantly (along τ and i transitions) until something is output. That something must be d.

- **No spurious output.** Consider the following formula:

$$
\begin{aligned}
&NoSpuOut'(i,o,a) = \\
&\quad (\nu X(b).[\tau]X(b) \wedge \forall c. \\
&\qquad ([\overline{c}](c = o \wedge ((\nu d \leftarrow X(b)) \vee (\exists d.d \neq b \wedge X(b))))) \wedge \\
&\qquad ([c](c = i \wedge \forall d.d \rightarrow (b = d \vee X(b)))))(a) \\
&NoSpuOut(i,o) = \forall a.NoSpuOut'(i,o,a)
\end{aligned}
$$

  The formula expresses, roughly, that an a must be input before it can be output. Observe that in the context of the π-calculus this gives an operational meaning to the notion of *authenticity:* it can be trusted that information emitted on o really was received along i.

- **No lost input.**

$$
\begin{aligned}
&EvOut(i,o,a) = \\
&\quad (\mu X(b).(<\tau>\top \vee <\overline{o}>\top \vee <i>\top) \wedge [\tau]X(b) \wedge \forall e. \\
&\qquad ([\overline{e}](e = o \wedge ((\nu c \leftarrow X(b)) \vee (\exists c.c \leftarrow X(b)) \vee (b \leftarrow \top)))) \wedge \\
&\qquad ([e](e = i \wedge \_ \rightarrow X(b))))(a)
\end{aligned}
$$

$$
\begin{aligned}
&NoLostInput(i,o) = \\
&\quad \nu X.[\tau]X \wedge \forall c. \\
&\qquad ([\overline{c}](c = o \wedge \_ \leftarrow X)) \wedge \\
&\qquad ([c](c = i \wedge \forall a.a \rightarrow X \wedge EvOut(i,o,a)))
\end{aligned}
$$

  For $NoLostInput(i,o)$ X must hold invariantly, and whenever an a is input then $EvOut(i,o,a)$ holds. The latter property expresses that some transition is enabled, and whatever transition is taken (among τ, i and o), either the transition was an output of a along o or else $EvOut(i,o,a)$ continues to hold. However, since we use a least fixed point eventually the former case must hold, so a is eventually output.

Example 4.4 (Natural Numbers) We wish to define the property $Nat(n)$ of "possessing a natural number object located at n". Keeping in mind the concrete representations of example 2.4, $Nat(n)$ should be expected to hold of a process P under the following circumstances:

- It should be possible to force P to synchronize along n, if not immediately then after an initial number of τ steps.
- No sequence of τ-labelled steps can disable an n-transition.
- All n-transitions take two arguments.
- Whenever fresh names s and z are provided along n the following properties will hold:
- At least one outgoing synchronisation along s or z is possible, but not both.
- Only unary outgoing synchronisations along s, or along z are possible.
- No sequence of internal steps can disable an s- or z-transition.
- Whenever a z transition takes place, the result is a process.
- Whenever an s-transition takes place a new name is output, serving as the location for the "predecessor of n".

We do not claim that these points unambiguously pin down the intended behaviour of "a natural number object" — in fact most of the points above leave room for debate. Resolving this we formalize the intuition in the π-μ-calculus.

We describe the behaviour of natural number objects as a state machine wrapped inside a least fixed point reflecting the well-foundedness property of natural numbers. We propose the following definition:

$$
\begin{aligned}
Nat_0(n,\phi) &\triangleq \mu Y.{<<}n{>>}\top \wedge \\
&\quad [\tau]Y \wedge \\
&\quad \forall b.[b](b = n \wedge \nu s \rightarrow \nu z \rightarrow \phi(s,z)) \wedge \\
&\quad \forall b.[\overline{b}]\bot \\
Nat_1(X)(s,z) &\triangleq \mu Z.\neg({<<}\overline{s}{>>}\top \wedge {<<}\overline{z}{>>}\top) \wedge \\
&\quad ({<<}\overline{s}{>>}\top \vee {<<}\overline{z}{>>}\top) \wedge \\
&\quad [\tau]Z \wedge \\
&\quad \forall b.[b]\bot \wedge \\
&\quad \forall b.[\overline{b}]((b = z \wedge isP) \vee (b = s \wedge \nu m \leftarrow X(m))) \\
Nat &\triangleq \mu X(n).Nat_0(n, Nat_1(X)) \qquad (6)
\end{aligned}
$$

The definition uses higher-order parameters in a manner which goes beyond the syntax as presented in the start of this section. However, at the expense

of a more monolithic and less readable notation it is quite easy to rewrite the definition to eliminate the higher-order abbreviations.

The idea of the definition is the following: We describe the behaviour of a natural number object as a sort of state machine formalizing a sort of natural number protocol. The state machine has two states, Nat_0 and Nat_1. Outside the state description is a least fixed point, used to reflect progress of entire protocol runs. The description of each state must bring out what actions are enabled, and how, and it must describe, since it will be used in a compositional manner, for each type of action, what the effect of performing such an action will be. In several case studies we have performed over the years we have found this sort of abstract state description very useful for proving properties of infinite state systems, and more examples will be given toward the end of this paper.

Now, suppose that P possesses a natural number object located at n. Then n is enabled and can be supplied with first a fresh s and then a fresh z. If n is supplied with a z or an s which is *not* fresh we can say little about the behaviour of the resulting system, as unintended internal communications can result. This is an important point, and it is the main reason why we have chosen to include fresh input as a primitive. Continuing the protocol either s or z is offered, but not both. If z is offered the result is a process and the protocol is terminated. If s is offered a new location m is output, and the agent will continue to behave as a natural number object located at m. Moreover, because of the least fixed point, the first option must eventually be selected, so the natural number object is ensured to be well-founded.

The natural number type given here is a candidate for the type of "one-shot", or ephemeral natural numbers. Expressing a property such as "n is the location of a natural number object from now until some future event takes place" is an easy embedding of *Nat* into an invariant. Many variations on the definition of *Nat* are possible. For instance we might not insist on well-foundedness, or *Nat* might be permitted to diverge.

5 Proof System, Modal Fragment

A *closed correctness assertion* is an assertion of the shape $\vdash P : \phi$ where P is an arbitrary process and ϕ is a π-μ-calculus sentence. For certain so-called finite-control agents that refrain from creating new processes dynamically (by avoiding parallel compositions in recursive contexts, or the replication operator) the problem of deciding validity of closed correctness assertions is decidable [4]. This is due to the fact that, up to choice of names and a little garbage collection, the state spaces of finite-control agents are finite [4]. Many interesting agents, however, fall outside the class of finite-control agents, including the natural number agents of example 2.4 and the unbounded buffers of example 2.3, because processes are created dynamically. In fact it is just this dynamic process creation capability that in conjunction with the capability of creating and communicating new names gives the π-calculus its remarkable expressive power, and it is also the feature that makes verification difficult.

Limits of Global State Exploration Approaches to verification which merely chase transitions and global states are unlikely to be very successful in proving interesting properties of non-finite-control agents. Consider for instance the unbounded buffer *Buf* of example 2.3. A temporal property that depends on *Buf* being continually able to input new data items will give rise to an unbounded state space quite trivially, as each input action gives rise to the creation of a new component process. But it may also be that despite being non-finite-control the state space is in fact bounded. As an example we may consider any process of the shape

$$P_n(b) = \nu a_n.COPY(a_n, b) \mid \nu a_{n-1}.\cdots \mid \nu a_0.SUCC(a_0, a_1) \mid ZERO(a_0) \quad (7)$$

which can only perform a bounded number of actions despite the dynamic process creation involved in the definition of $COPY$. However, the interesting correctness property of $COPY$ is less the collection of n facts that (for instance) $\nu b.(COPY(b, c) \mid P_n(b))$ represents a natural number located at c, but rather the fact that $COPY$ is type correct, ie. that for *any* x which represents a natural number located at b, $\nu b.(COPY(b, c) \mid x)$ represents a natural number located at c. But this latter assertion is not a closed correctness assertion, and it is not within the scope of model checking techniques such as that of [4] that explore global state spaces since the agent expression being predicated is open.

Open Correctness Assertions We thus need to address more general *open correctness assertions* that allow correctness properties $P : \phi$ to be made conditional upon properties of the parameters of P (such as: x represents a natural number located at b). This can give a handle on dynamic process creation in the following way: Suppose the goal is to prove a correctness assertion of the shape

$$\vdash P : \phi. \quad (8)$$

Assume that P involves dynamic process creation and that through a number of steps the proof goal (8) is "reduced" to one the shape

$$\vdash P \mid Q : \psi \quad (9)$$

For instance P may be the agent *Buf* i o of example 2.3. The idea now is to apply a cut, guessing a property γ to hold of P, and then reduce (9) to the proof goals

$$\vdash P : \gamma \quad (10)$$

and

$$x : \gamma \vdash x \mid Q : \psi \quad (11)$$

representing the assertion that ψ holds on $x \mid Q$ conditional upon γ holding on x. Say, for instance, that we can choose $\gamma = \phi$ which turns out to be the case in many examples. Then the proof goal (10) is an instance of (8) and it may consequently be that the subgoal (10) for this reason can be discharged. If in turn Q does not involve dynamic process creation the problem (of dealing with this feature) may have been resolved, and if it does we will want to iterate the approach, keeping in mind that we now have to deal with more general open correctness assertions.

5.1 Basic Judgments

In [5] this idea was worked out for CCS. A delicate issue in generalising the approach to the π-calculus is how to deal with private names, name generation, and scope extrusion. In general one will wish to verify properties of a process $\nu a.P(x)$ relative to a property ψ of x. ψ must be allowed to depend on a, Consider for instance the property that $\nu b.(COPY\ b\ c \mid x)$ is a natural number located at c given that x is a natural number located at b. But if we take alpha-conversion for granted then $\nu b.(COPY\ b\ c \mid x)$ should be indistinguishable from $\nu b'.(COPY\ b'\ c \mid x)$ provided no name clashes arise, and the dependency upon b is lost. We thus need a mechanism to "freeze" b to extend its scope to cover also formulas to the left of the turnstile. In [1] we suggested annotating the turnstile with a "restriction set" s for this purpose. We are following this suggestion here. Thus judgments take the more general form $x : \psi \vdash^s P : \phi$ where s is a finite set of names with a scope extending over both P and ψ, but *not* over ϕ. In fact a very similar annotation was introduced already by Stirling [18]. Here, however, the annotation has a somewhat deeper function, due to the richer name discipline of the π-calculus.

Term variables and open terms Before defining formally the notion of judgment and its semantics observe that we need to extend the basic syntax of the π-calculus to open terms. We use x, y, and z as term variables (to range over processes, abstractions, and concretions). Terms in $\mathcal{P}$, $\mathcal{A}$, or $\mathcal{C}$ may from now on be open, i.e. involve term variables. A *closed term* will be a term which does not contain term variables (but it may contain free names). We use E to range over $\mathcal{P} \cup \mathcal{A} \cup \mathcal{C}$.

The consideration of open terms also leads us to extend the π-calculus syntax slightly, by allowing the parallel composition operator to be applied not only to processes but also to abstractions and concretions (cf. [10]). This is done by rewriting (where symmetric versions are assumed to be added implicitly):

$$\begin{array}{rcl} ((\vec{b})A) \mid P & \rightarrow & (\vec{b})(A \mid P) \\ (\nu\vec{c_1}.\langle\vec{c_2}\rangle C) \mid P & \rightarrow & \nu\vec{c_1}.\langle\vec{c_2}\rangle(C \mid P) \\ ((\vec{b})A) \mid (\nu\vec{c_1}.\langle\vec{c_2}\rangle C) & \rightarrow & \nu\vec{c_1}.(A\{\vec{c_2}/\vec{b}\} \mid C) \end{array}$$

We assume that symmetric versions of these rewrite rules are added implicitly, and appeal to alpha-conversion to prevent variable capture as ever.

Definition 5.1 (Basic Judgments, Validity) A *basic judgment* is an expression of the form

$$\Gamma \vdash^s E : \Delta \tag{12}$$

where

1. $\Gamma = \{x_1 : \phi_1, \ldots, x_n : \phi_n\}$ is a finite set (of *assumptions*) such that the ϕ_i are sentences, $1 \leq i \leq n$,
2. $s = \{a_1, \ldots, a_k\}$ is a finite set of names, the *restriction set*,

3. $\Delta = \{\psi_1, \ldots, \psi_m\}$ is a finite set of sentences, and

4. E is an open term.

An agent variable x occurs freely in $\Gamma \vdash^s E : \Delta$ if either x occurs freely in E or Γ contains an assumption on x, ie. an assumption of the shape $x : \phi$.

For the semantics name interpretations η are extended to general substitutions by mapping both names to names and term variables to closed terms. In doing so the sets of names and of term variables are assumed to be distinct. We can then extend $\nu s.\eta$ to substitutions by restricting η to names. Then, the judgment $\Gamma \vdash^s E : \Delta$ is *valid* (or *true*, written $\Gamma \models^s E : \Delta$) if for all substitutions η, if $\models_{\nu s.\eta} \eta(x_i) : \phi_i$ for all $i : 1 \leq i \leq n$, then $\models_\eta \nu s.(E\eta) : \psi_j$ for some $j : 1 \leq j \leq m$.

Notation For sets such as Γ, s, and Δ we use a standard sequence-like notation, writing eg. $\Gamma, x_1 : \phi_1, x_2 : \phi_2$ in place of $\Gamma \cup \{x_1 : \phi_1, x_2 : \phi_2\}$, or $\Gamma \vdash^{()} A : \Delta_1, \Delta_2$ in place of $\Gamma \vdash^{\emptyset} A : \Delta_1 \cup \Delta_2$. For a basic judgment $\Gamma, x : \phi \vdash^s E : \Delta$, if ϕ is elementary (so that the holding of ϕ does not depend on x), we allow the judgment to be abbreviated by $\Gamma, \phi \vdash^s E : \Delta$.

Restrictions Sets and Scoping The point already made concerning scope of restriction sets deserves to be reiterated. Observe that in a judgment of the shape $x : \phi \vdash^a E : \psi$ both ϕ, E and ψ may mention a. Let η be an arbitrary name interpretation, and suppose that $\models_{\nu a.\eta} \eta(x) : \phi$. Then occurrences of a in ϕ refer, because of the use of the name interpretation $\nu a.\eta$, to occurrences of a in $\eta(x)$. Moreover, no other name occurring in ϕ can be confused with a in $\eta(x)$. In forming $\nu a.(E\eta)$ the a's in E and the a's in $\eta(x)$ are identified (they are equal). An a occurring in ψ, on the other hand, can, through η, be identified with or distinguished from any name occurring freely in $\nu a.(E\eta)$, but in that agent a itself is bound. So the scope of a in $x : \phi \vdash^a E : \psi$ extends to E and ϕ, but *not* to ψ. This is reflected in the proof system below by the rule (Alpha).

5.2 A Proof System for the Modal Fragment

We now turn to the problem of proving validity of basic judgments, restricting attention for now to the modal fragment. A proof system will consist of a number of clearly discernible parts:

1. A group of structural rules governing aspects like the introduction and use of assumptions.

2. A group of logical rules to deal with connectives like conjunction, disjunction, and the quantifiers.

3. A group of rules for name equality and inequality.

4. A group of rules for the process modalities $<\alpha>$ and $[\alpha]$.

5. A group of rules for the input/output modalities $a \to \phi$, $a \leftarrow \phi$, $\nu a \to \phi$ and $\nu a \leftarrow \phi$.

The first three groups of rules are based on a standard sequent-style formalisation of first-order logic with equality. The adaptation is not completely trivial, however, due to the presence of agent terms and variables, and restriction sets. An important and delicate issue concerns the choice of rules to include as primitive. Our strategy here is to include only rules that are needed in a completeness argument, but it is probably useful to bear in mind that this is far less clearcut and definitive a criterion than may be thought at first glance, and other less tangible criteria like "elegance", or "orthogonality" are important too in the detailed formulation of rules.

We go through each group of rules in turn.

5.3 Structural Rules

We first have a minimal set of rules for introducing and applying assumptions, namely the identity, weakening, and the cut, bearing in mind that in general these rules are affected by the presence of restriction sets:

$$\text{(I)} \quad \frac{\cdot}{\Gamma, x : \phi \vdash^{()} x : \phi, \Delta}$$

$$\text{(W-L)} \quad \frac{\Gamma \vdash^{s} E : \Delta}{\Gamma, x : \phi \vdash^{s} E : \Delta} \qquad \text{(W-R)} \quad \frac{\Gamma \vdash^{s} E : \Delta}{\Gamma \vdash^{s} E : \psi, \Delta}$$

As the Γ, s, and Δ are sets no rules for contraction and permutation are needed. We comment on the need for weakening later. The cut rule comes in three flavours. The first, (Cut-1), is essential to accomodate reasoning on agent structure and it is not in general eliminable.

$$\text{(Cut-1)} \quad \frac{\Gamma, s \mathit{\ fresh} \vdash^{()} P : \phi \quad \Gamma, x : \phi \vdash^{s} E : \Delta}{\Gamma \vdash^{s} E\{P/x\} : \Delta} \quad (x \notin \Gamma)$$

We here introduce two new pieces of shorthand, writing $x \notin \Gamma$ for the condition that Γ does not contain an assumption on x, and s *fresh* for an elementary formula expressing that all names in s are distinct from each other, and from any name occurring in Γ or Δ.

To illustrate the complications involved in the first cut rule assume that the judgments $\Gamma, s \mathit{\ fresh} \vdash^{()} P : \phi$ and $\Gamma, x : \phi \vdash^{s} E : \Delta$ are valid, and that a substitution η is given such that all assumptions in Γ are validated for the name interpretation $\nu s.\eta$. The substitution $\nu s.\eta$ validates also the condition s *fresh*, so we can conclude that $\models_{\nu s.\eta} P\eta : \phi$. Now x does not occur in Γ so η can be extended to the substitution $\eta' = \eta\{P\eta/x\} = \{P/x\}\eta$ which validates both Γ and $x : \phi$ for the restriction set s. But then $\models_{\eta} \nu s.(E\{P/x\})\eta) : \Delta$ as required.

The second cut rule will be needed when we come to consider logical fixed points and discharge by loop termination by providing garbage collection of

restrictions that are no longer used. For the modal fragment, however, (Cut-2) is admissible, as can be seen from the completeness proof below.

$$\text{(Cut-2)} \quad \frac{\Gamma, s_2\ \mathit{fresh} \vdash^{s_1} E : \phi \qquad \Gamma, x : \phi \vdash^{s_2} x : \Delta}{\Gamma \vdash^{s_1, s_2} E : \Delta} \quad (x \notin \Gamma, s_1 \cap s_2 = \emptyset)$$

Here s_2 *fresh* is used in the sense of requiring also names in s_2 to be distinct from names in s_1. To see the soundness of this rule assume $\Gamma, s_2\ fresh \models^{s_1} E : \phi$, $\Gamma, x : \phi \models^{s_2} x : \psi$, and $s_1 \cap s_2 = \emptyset$. Let η be given such that $\eta(x_i)$ is a process for all $i : 1 \leq i \leq n$.. Assume that $\eta(x) \in \|\phi_i\|(\nu s_1.\nu s_2.\eta)$ whenever $x = x_i$. By the first assumption, $\nu s_1.(P\eta) \in \|\phi\|(\nu s_2.\eta)$. But then $\nu s_2.\nu s_1.(P\eta) = \nu s_1.\nu s_2.(P\eta) \in \|\psi\|\eta$ as desired.

The third cut rule is the following:

$$\text{(Cut-3)} \quad \frac{\Gamma \vdash^{s} E : \phi, \Delta \qquad \Gamma, y : \phi \vdash^{()} y : \Delta}{\Gamma \vdash^{s} E : \Delta}$$

where y does not occur in Γ. For the modal fragment (Cut-3) is admissible. We conjecture that this is not so in general. However for $s = ()$ the rule (Cut-3) is derivable quite easily using the logical rules introduced in the following section.

Finally we need the following rule to reflect the scoping rule for restriction sets:

$$\text{(Alpha)} \quad \frac{\Gamma\sigma \vdash^{s\sigma} E\sigma : \Delta}{\Gamma \vdash^{s} E : \Delta}$$

where $\sigma : s \to \mathcal{G}$ is injective, the range of σ is disjoint from $(fn(\Gamma) \cup fn(E) \cup fn(\Delta)) - s$, and the postfixing of σ is the extension of σ to agents, restriction sets, and assumption sets Γ. Other substitution rules such as injective substitutions of unrestricted names, or of term variables, are admissible.

5.4 Logical Rules

As a first approximation we require standard rules for introducing elementary connectives to the left and to the right.

$$\text{($\wedge$-L)} \quad \frac{\Gamma, x : \phi, x : \psi \vdash^{s} E : \Delta}{\Gamma, x : \phi \wedge \psi \vdash^{s} E : \Delta} \qquad \text{($\wedge$-R)} \quad \frac{\Gamma \vdash^{s} E : \phi, \Delta \qquad \Gamma \vdash^{s} E : \psi, \Delta}{\Gamma \vdash^{s} E : \phi \wedge \psi, \Delta}$$

$$\text{($\vee$-L)} \quad \frac{\Gamma, x : \phi \vdash^{s} E : \Delta \qquad \Gamma, x : \psi \vdash^{s} E : \Delta}{\Gamma, x : \phi \vee \psi \vdash^{s} E : \Delta}$$

$$\text{($\vee$-R)} \quad \frac{\Gamma \vdash^{s} E : \phi, \psi, \Delta}{\Gamma \vdash^{s} E : \phi \vee \psi, \Delta}$$

$$\text{($\forall$-L)} \quad \frac{\Gamma, x : \phi\{b/a\} \vdash^{s} E : \Delta}{\Gamma, x : \forall a.\phi \vdash^{s} E : \Delta} \qquad \text{($\forall$-R)} \quad \frac{\Gamma \vdash^{s} E : \phi, \Delta}{\Gamma \vdash^{s} E : \forall a.\phi, \Delta} \quad (a \text{ fresh})$$

$$\text{($\exists$-L)} \quad \frac{\Gamma, x : \phi \vdash^{s} E : \Delta}{\Gamma, x : \exists a.\phi \vdash^{s} E : \Delta} \quad (a \text{ fresh}) \qquad \text{($\exists$-R)} \quad \frac{\Gamma \vdash^{s} E : \phi\{b/a\}, \Delta}{\Gamma \vdash^{s} E : \exists a.\phi.\Delta}$$

In the rules $\forall$-R and $\exists$-L we use *a fresh* as a shorthand for the condition that a is not free in the conclusion judgment and, in the case of $\exists$-L, neither does a occur in s.

Since the logic is closed under classical negation we further need rules e.g. to reflect that contradictory assumptions can be made governing the same agent variable. We suggest the following two rules, left and right dilemma:

$$\text{(D-L)} \quad \frac{\Gamma, x : \phi \vdash^s E : \Delta \quad \Gamma, x : \neg\phi \vdash^s E : \Delta}{\Gamma \vdash^s E : \Delta}$$

$$\text{(D-R)} \quad \frac{\Gamma \vdash^s E : \phi, \Delta \quad \Gamma \vdash^s E : \neg\phi, \Delta}{\Gamma \vdash^s E : \Delta}$$

Two further rules are needed to reflect the fact that elementary formulas do not depend on the agent term being predicated:

$$\text{(E-L)} \quad \frac{\Gamma, y : \phi \vdash^s E : \Delta}{\Gamma, x : \phi \vdash^s E : \Delta} \qquad \text{(E-R)} \quad \frac{\Gamma \vdash^s E : \phi}{\Gamma \vdash^s E' : \phi}$$

where both rules require that ϕ is elementary.

Lemma 5.2 *The following rules are derivable:*

$$\neg\text{-L} \quad \frac{\Gamma \vdash^() x : \phi, \Delta}{\Gamma, x : \neg\phi \vdash^() x : \Delta} \qquad \neg\text{-R} \quad \frac{\Gamma, x : \phi \vdash^() x : \Delta}{\Gamma \vdash^() x : \neg\phi, \Delta}$$

PROOF: Easy derivations using weakening and dilemma. □

5.5 Rules for Equality and Inequality

For equality and inequality we suggest four axioms and one rule of inference. The four axioms, first, express the following properties:

1. Names are equal to themselves.
2. Even possibly restricted names are equal to themselves.
3. A name can not be identified with a name in the restriction set unless it is textually identical.
4. There are infinitely many names.

The addition is a rule of substitution of equals for equals.

$$\text{(REFL)} \quad \frac{\cdot}{\Gamma \vdash^s E : a = a, \Delta}$$

$$\text{(IRR)} \quad \frac{\cdot}{\Gamma, a \neq a \vdash^s E : \Delta}$$

$$\text{(NEW1)} \quad \frac{\cdot}{\Gamma, x : a = b \vdash^{s,a} E : \Delta} \quad (a \neq b)$$

$$(\textsc{Infty}) \quad \frac{\cdot}{\Gamma \vdash^s E : \exists b.b \neq a_1 \wedge \cdots \wedge b \neq a_n, \Delta}$$

$$(\textsc{Subst}) \quad \frac{\Gamma\{b/c\} \vdash^s E\{b/c\} : \Delta\{b/c\}}{\Gamma\{a/c\}, a = b \vdash^s E\{a/c\} : \Delta\{a/c\}} \quad (a, b \notin s)$$

To see the soundness of (New1) observe that it can not simultaneously be the case that $\models_{\nu s.\nu a.\eta} \eta(x) : a = b$ and that a and b are distinct names. Observe also that the rule (Irr) is needed only for $a \in s$. We note the derivability of a number of useful proof rules.

Lemma 5.3 *The following rules are derivable:*

$$(\text{Eq-I}) \quad \frac{\cdot}{\Gamma, a = b \vdash^s E : a = b, \Delta} \quad (a, b \notin s)$$

$$(\textsc{Elem-I}) \quad \frac{\cdot}{\Gamma, \phi \vdash^s E : \phi, \Delta} \quad (\phi \text{ elementary}, \mathrm{fn}(\phi) \cap s = \emptyset)$$

$$(\textsc{Ineq}) \quad \frac{\Gamma\{b/a\} \vdash^s E\{b/a\} : \Delta\{b/a\}}{\Gamma \vdash^s E : a \neq b, \Delta} \quad (a, b \notin s)$$

$$(\textsc{Sym1}) \quad \frac{\Gamma \vdash^s E : b = a, \Delta}{\Gamma \vdash^s E : a = b, \Delta}$$

$$(\textsc{Tr}) \quad \frac{\Gamma \vdash^s E : a = b, \Delta \quad \Gamma \vdash^s E : b = c, \Delta}{\Gamma \vdash^s E : a = c, \Delta}$$

$$(\textsc{Sym2}) \quad \frac{\Gamma \vdash^s E : a \neq b, \Delta}{\Gamma \vdash^s E : b \neq a, \Delta}$$

$$(\textsc{Dist}) \quad \frac{\Gamma \vdash^s E : a \neq b, \Delta \quad \Gamma \vdash^s E : b = c, \Delta}{\Gamma \vdash^s E : a \neq c, \Delta}$$

Proof: (Eq-I): Use (Subst) and (Refl).

(Elem-I): Use Eq-I) and the dilemma rules along with the logical rules and structural induction in ϕ.

(Ineq): Use (D-l), weakening, (Elem-I), and (Subst).

(Sym1): This case is slightly more delicate than the previous ones. We prove $\Gamma \vdash^s E : b = a, \Delta$ given a proof of $\Gamma \vdash^s E : a = b, \Delta$. First observe that $\Gamma \vdash^s a = b, b = a, \Delta$ by weakening. Then observe that $\Gamma \vdash^{()} b = b, \Delta$ so that $\Gamma, a = b \vdash^{()} b = a, \Delta$ by (Subst). Then the proof is complete by (Cut-3).

The remaining derivations are easy exercises. □

5.6 Rules for Transition Modalities

We then arrive at the rules for the modalitites $<l>$ and $[l]$. Proof goals in this case have the following shape:

$$\Gamma \vdash^s E : (l)\phi \tag{13}$$

where (l) is used as a wildcard ranging over $<l>$ and $[l]$. Observe that this shape is not quite as general as we would wish it to be. Rather we would want to permit sets of modal formulas to the right of the turnstile instead of the single formula permitted here. The restricted format (13) is chosen for the following reasons:

1. Modal proof rules for more general multi-conclusioned judgments would become unduly complicated.

2. The restricted format suffices for both the examples and the weak completeness results which we present here.

The transition capabilities of E in (13) are determined by the structure of E according the operational semantics given earlier, and according to the assumptions made in Γ on variables occurring freely in E. As the operational semantics is determined by induction in the structure of E it is hardly surprising that in general the number of rules governing proof goals of the shape $\Gamma \vdash^s E : (\alpha)\phi_1, \ldots, (\alpha)\phi_n$ depends on the number of primitive operators, and for each operator, the number of operational semantics rules of section 2 determining its behaviour.

We then proceed by induction in the structure of the right hand agent term being predicated to give rules that show how (typically modal, but in some cases quite general) properties of an agent with a specific outermost connective can be inferred in terms of properties of its immediate constituents.

Term Variables The case for E a variable corresponds to the monotonicity rule familiar from sequent-style formalisations of modal logic:

$$(<l_\tau>\text{-MON}) \quad \frac{\Gamma, y : \phi, x : \phi_1, \ldots, y : \phi_n \vdash^s y : \psi_1, \ldots, \psi_m}{\Gamma, x : <l_\tau>\phi, x : [l_\tau]\phi_1, \ldots, x : [l_\tau]\phi_n \vdash^s x : <l_\tau>\psi_1, \ldots, <l_\tau>\psi_m}$$

$$([l_\tau]\text{-MON}) \quad \frac{\Gamma, y : \phi_1, \ldots, y : \phi_n \vdash^s y : \psi, \psi_1, \ldots, \psi_m}{\Gamma, x : [l_\tau]\phi_1, \ldots, x : [l_\tau]\phi_n \vdash^s x : [l_\tau]\psi, <l_\tau>\psi_1, \ldots, <l_\tau>\psi_m}$$

For both rules we require that $y \notin \Gamma$. With this proviso the rules are clearly sound.

Nil

$$([l_\tau]\text{-NIL}) \quad \frac{\cdot}{\Gamma \vdash^s 0 : [l_\tau]\phi}$$

Summation

$$<l_\tau>\text{-PLUS} \quad \frac{\Gamma \vdash^s P_i : <l_\tau>\phi}{\Gamma \vdash^s P_1 + P_2 : <l_\tau>\phi, \Delta} \quad (i \in \{1,2\})$$

$$([l_\tau]\text{-PLUS}) \quad \frac{\Gamma \vdash^s P_1 : [l_\tau]\phi \quad \Gamma \vdash^s P_2 : [l_\tau]\phi}{\Gamma \vdash^s P_1 + P_2 : [l_\tau]\phi}$$

Prefixing

$$((\tau)\text{-TAU}) \quad \frac{\Gamma \vdash^s P : \phi}{\Gamma \vdash^s \tau.P : (\tau)\phi} \qquad ([\tau]\text{-ACT}) \quad \frac{\cdot}{\Gamma \vdash^s l.E : [\tau]\phi}$$

$$([l]\text{-TAU}) \quad \frac{\cdot}{\Gamma \vdash^s \tau.P : [l]\phi} \qquad ([a]\text{-}\overline{b}) \quad \frac{\cdot}{\Gamma \vdash^s \overline{b}.C : [a]\phi}$$

$$([\overline{a}]\text{-}b) \quad \frac{\cdot}{\Gamma \vdash^s b.A : [\overline{a}]\phi}$$

$$(<l>\text{-ACT}) \quad \frac{\Gamma \vdash^s E : \phi \quad \Gamma \vdash^s E : a_1 = a_2}{\Gamma \vdash^s \hat{a_1}.E : <\hat{a_2}>\phi} \quad (l_1 \notin s)$$

$$([l]\text{-ACT}) \quad \frac{\Gamma, a_1 = a_2 \vdash^s E : \phi}{\Gamma \vdash^s \hat{a_1}.E : [\hat{a_2}]\phi} \quad (a_2 \notin s)$$

In the two last rules the notation $\hat{a}$ is used to indicate overbarring which is optional in the sense that either both transition labels indicated are overbarred, or else none is.

The rule ($[l]$-ACT) reveals where (ALPHA) is required: To show $\vdash^a a.A : [a]\bot$ (which is a valid judgment for any A) first we need to use (ALPHA) to rename the restricted a, then use ($[l]$-ACT) to reduce to a goal of the shape $b = a \vdash^b A : \bot$, which is then resolved by (NEW1).

Parallel Composition The rules for parallel composition are shown on table 3. The rule ($<l>$-PAR) comes with a symmetric version. All rules for $|$ are marked $*$ indicating that they are subject to the side-condition that x and y are fresh (ie. do not appear free in the conclusion judgment), and the typing formulas $\mathcal{S}_1$ and $\mathcal{S}_2$ (where they appear) are complementary. The typing formulas have the important role of matching input and output. This is evident in the rule ($<\tau>$-PAR). In ($[\tau]$-PAR) the role is implicit: The rule requires an ancillary rule schema

$$(\text{AR}) \quad \frac{\Gamma \vdash^s E : \mathcal{S}_1 \quad \Gamma \vdash^s E : \mathcal{S}_2}{\Gamma \vdash^s F : \phi} \quad (S_1 \neq S_2)$$

to ensure that inputs are matched with outputs of the proper sort.

To see the soundness of eg. ($<\tau>$-PAR) assume that the antecedents of that rule are valid. Assume also that a substitution η is given such that $\models_{\nu s.\eta} \eta(x_i) : \phi_i$ for all $i : 1 \leq i \leq n$. Then $\models E\eta : <l>\phi$ and $\models F\eta : <\overline{l}>\psi$. Thus $E\eta$ and $F\eta$ are processes, and there are E' and F' such that $E\eta \xrightarrow{l} E'$ and $F\eta \xrightarrow{\overline{l}} F'$.

$(<l>\text{-PAR})^*$:

$$\frac{\Gamma, s\ \mathit{fresh} \vdash^{()} F : \mathit{isP} \quad \Gamma, s\ \mathit{fresh} \vdash^{()} E : <l>\phi \quad \Gamma, x : \phi \vdash^{s} x \mid F : \gamma}{\Gamma \vdash^{s} E \mid F : <l>\gamma}$$

$(<\tau>\text{-PAR})^*$:

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} E : <l>\phi \quad \Gamma, s\ \mathit{fresh} \vdash^{()} F : <\bar{l}>\psi \\ \Gamma, s\ \mathit{fresh}, x : \phi \vdash^{()} x : \mathcal{S}_1 \quad \Gamma, s\ \mathit{fresh}, y : \phi \vdash^{()} y : \mathcal{S}_2 \\ \Gamma, x : \phi, y : \psi \vdash^{s} x \mid y : \gamma\end{array}}{\Gamma \vdash^{s} E \mid F : <\tau>\gamma}$$

$([l]\text{-PAR})^*$:

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} E : \mathit{isP} \quad \Gamma, s\ \mathit{fresh} \vdash^{()} F : \mathit{isP} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} E : [l]\phi_1 \ \cdots \ \Gamma, s\ \mathit{fresh} \vdash^{()} E : [l]\phi_n \\ \Gamma, s\ \mathit{fresh} \vdash^{()} F : [l]\psi_1 \ \cdots \ \Gamma, s\ \mathit{fresh} \vdash^{()} F : [l]\psi_m \\ \Gamma, x : \phi_1, \ldots, x : \phi_n \vdash^{s} x \mid F : \gamma \quad \Gamma, y : \psi_1, \ldots, y : \psi_n \vdash^{s} E \mid y : \gamma\end{array}}{\Gamma \vdash^{s} E \mid F : [l]\gamma}$$

$([\tau]\text{-PAR})^*$:

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} E : \mathit{isP} \quad \Gamma, s\ \mathit{fresh} \vdash^{()} F : \mathit{isP} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} E : [\tau]\phi_{1,1} \ \cdots \ \Gamma, s\ \mathit{fresh} \vdash^{()} E : [\tau]\phi_{1,n_1} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} F : [\tau]\psi_{1,1} \ \cdots \ \Gamma, s\ \mathit{fresh} \vdash^{()} F : [\tau]\psi_{1,m_1} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} E : \forall b.[b]\phi_{2,1} \ \cdots \ \Gamma, s\ \mathit{fresh} \vdash^{()} E : \forall b.[b]\phi_{2,n_2} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} E : \forall b.[\bar{b}]\phi_{3,1} \ \cdots \ \Gamma, s\ \mathit{fresh} \vdash^{()} E : \forall b.[\bar{b}]\phi_{3,n_3} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} F : \forall b.[b]\psi_{2,1} \ \cdots \ \Gamma, s\ \mathit{fresh} \vdash^{()} F : \forall b.[b]\psi_{2,m_2} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} F : \forall b.[\bar{b}]\psi_{3,1} \ \cdots \ \Gamma, s\ \mathit{fresh} \vdash^{()} F : \forall b.[\bar{b}]\psi_{3,m_3} \\ \Gamma, x : \phi_{1,1}, \ldots, x : \phi_{1,n_1} \vdash^{s} x \mid F : \gamma \\ \Gamma, y : \psi_{1,1}, \ldots, y : \psi_{1,m_1} \vdash^{s} E \mid y : \gamma \\ \Gamma, x : \phi_{2,1}, \ldots, x : \phi_{2,n_2}, x : \mathit{isA}, y : \psi_{3,1}, \ldots, y : \psi_{3,m_3}, y : \mathit{isCf} \vdash^{s} x \mid y : \gamma \\ \Gamma, x : \phi_{2,1}, \ldots, x : \phi_{2,n_2}, x : \mathit{isA}, y : \psi_{3,1}, \ldots, y : \psi_{3,m_3}, y : \mathit{isCb} \vdash^{s} x \mid y : \gamma \\ \Gamma, x : \phi_{3,1}, \ldots, x : \phi_{3,n_3}, x : \mathit{isCf}, y : \psi_{2,1}, \ldots, y : \psi_{2,m_2}, y : \mathit{isA} \vdash^{s} x \mid y : \gamma \\ \Gamma, x : \phi_{3,1}, \ldots, x : \phi_{3,n_3}, x : \mathit{isCb}, y : \psi_{2,1}, \ldots, y : \psi_{2,m_2}, y : \mathit{isA} \vdash^{s} x \mid y : \gamma\end{array}}{\Gamma \vdash^{s} E \mid F : [\tau]\gamma}$$

Table 3: Proof rules for parallel composition vs. transition modalities

Moreover we can assume that E' and F' have complementary arities by the third and fourth antecedents. Moreover, $\models_{\nu s.\eta} \nu s.(x \mid y\eta\{E'/x\}\{F'/y\}) : \gamma$. But then $\models_{\nu s.\eta} E \mid F\eta : <\tau>\gamma$ as desired.

The rule ($[\tau]$-PAR) is at first sight ridiculously complex. On closer inspection, however, we argue that the rule merely brings out the quite complex modal behaviour of π-calculus parallel composition, and thus the complexity is inherent in the problem rather than due to the specificities of our formalisation. This is not to say that simpler formulations can not be found. In fact this may very well be possible, for instance by appealing directly to the operational semantics transition relation in a way which we have chosen *not* to do. However, we do believe quite strongly that a truly *compositional* and modal analysis of π-calculus parallel composition will have to perform the sort of quite convoluted case analysis brought out by the ($[\tau]$-PAR) rule.

Conditional

$$(\textsc{Cond}) \quad \frac{\Gamma, a = b \vdash^s P : \phi \quad \Gamma, a \neq b \vdash^s Q : \phi}{\Gamma \vdash^s \textbf{if } a = b \textbf{ then } P \textbf{ else } Q : \phi}$$

Restriction

$$(\textsc{New2}) \quad \frac{\Gamma \vdash^{s,a} E : \phi}{\Gamma \vdash^s \nu a.E : \phi} \quad (a \text{ fresh})$$

Identifiers

$$(\textsc{Fix}) \quad \frac{\Gamma \vdash^s P\{\vec{b}/\vec{a}\} : \phi}{\Gamma \vdash^s D(\vec{b}) : \phi} \quad D(\vec{a}) \triangleq P$$

Of these final three rules the rule (NEW2) is the least trivial. Assume that a is fresh for $\Gamma \vdash^s \nu a.E : \phi$, and that $\Gamma \models^{s,a} E : \phi$. Let η be given such that $\eta(x) \in \|\phi_i\|(\nu s.\eta)$ whenever $x = x_i$. Pick now a b which is fresh and not in free in any $\eta(x_i)$. Let $\eta'(x_i) = \eta(x)\{b/a\}$. Then $\eta'(x) \in \|\phi_i\|(\nu b.\nu s.\eta)$, so, by the assumption, and substituting b uniformly for a, $\nu s.\nu b.E\eta' \in \|\phi\|\eta$. But then $\nu s.(\nu a.E)\eta \in \|\phi\|\eta$ too.

Of all the rules for process modalities clearly the rule ($[\tau]$-PAR) is the most complex. It is nonetheless quite intuitive. To prove a property of the shape $[\tau]\phi$ of a parallel composition $A \mid B$ we have to describe in sufficient detail, by formulas $\phi_{i,j}$ and $\psi_{i,j}$, the properties of A and B after a transition has been performed. For almost all a, $\phi_{2,j}(a)$ and $\psi_{2,j}(a)$ will be false. The formulas $\phi_{i,j}$ and $\psi_{i,j}$ have to be shown to hold for A or B in the prescribed fashion, and they have to be shown to compose in the correct way, given that synchronisations must result in agents of complementary types.

Many variations can be played on the formulation of these rules. As an example we consider a version of ($[l]$-PAR) given in [5] (for CCS, so typing formulas and restriction sets were not needed there).

Proposition 5.4 *The following rule is derivable where x and y do not occur free in Γ:*

$$([l]\text{-PAR'}) \quad \frac{\begin{array}{c}\Gamma, x:\phi_1, y:\psi_2 \vdash^s x \mid y:\gamma \\ \Gamma, x:\phi_2, y:\psi_1 \vdash^s x \mid y:\gamma\end{array}}{\Gamma, x:\phi_1 \wedge [l]\phi_2, y:\psi_1 \wedge [l]\psi_2 \vdash^s x \mid y:[l]\gamma}$$

PROOF: The proof goal is

$$\Gamma, x:\phi_1 \wedge [l]\phi_2, y:\psi_1 \wedge [l]\psi_2 \vdash^s x \mid y:[l]\gamma. \tag{14}$$

We get the following list of 6 subgoals:

$$\Gamma, s\ \mathit{fresh}, x:\phi_1 \wedge [l]\phi_2, y:\psi_1 \wedge [l]\psi_2 \vdash^{()} x:\mathit{isP} \tag{15}$$
$$\Gamma, s\ \mathit{fresh}, x:\phi_1 \wedge [l]\phi_2, y:\psi_1 \wedge [l]\psi_2 \vdash^{()} y:\mathit{isP} \tag{16}$$
$$\Gamma, s\ \mathit{fresh}, x:\phi_1 \wedge [l]\phi_2, y:\psi_1 \wedge [l]\psi_2 \vdash^{()} x:[l]\phi_2 \tag{17}$$
$$\Gamma, s\ \mathit{fresh}, x:\phi_1 \wedge [l]\phi_2, y:\psi_1 \wedge [l]\psi_2 \vdash^{()} y:[l]\psi_2 \tag{18}$$
$$\Gamma, x:\phi_1 \wedge [l]\phi_2, y:\psi_1 \wedge [l]\psi_2, x':\phi_2 \vdash^s x' \mid y:\gamma \tag{19}$$
$$\Gamma, x:\phi_1 \wedge [l]\phi_2, y:\psi_1 \wedge [l]\psi_2, y':\psi_2 \vdash^s x \mid y':\gamma \tag{20}$$

To prove (15) and (16) we use a few logical and equational rules along with ($[l_\tau]$-MON). Subgoals (17) and (18) are trivial. To prove (19) and (20) apply weakening to eliminate the "old" version of x (or y), and then apply ($\wedge$-L) to arrive at the subgoal $\Gamma, x':\phi_2, y:\psi_1 \vdash^s x' \mid y:[\tau]\gamma$ (for (19)) as desired.

A small but important difference between the rules ($[l]$-PAR') and ($[l]$-PAR) is that in ($[l]$-PAR) (as elsewhere) we permit multiple assumptions on term variables. This is inessential for the modal fragment (the version with multiple assumptions can easily be derived), but when we come to consider recursive formulas the distinction will turn out to be more significant.

5.7 Rules for Input/Output Modalities

Except for two rules needed for conversion between the free and bound input modalities, the rules for input and output modalities follow the pattern established for the process modalities in the previous section. As there introduced variables like x and y below are subject to the side condition that they do not appear in Γ.

Term Variables The rules for term variables are shown on table 4. The side-condition a fresh in ($\nu \rightarrow$-MON) and ($\nu \leftarrow$-MON) means, as before, that a does not occur freely in the conclusion, nor is a a member of s.

The most delicate of the motonicity rules is ($\nu \leftarrow$-MON). To see that it is sound assume for simplicity that Γ is empty. Let η and C be given such that $\models_{\nu s.\eta} C : \nu a \leftarrow \phi_i$ for all $i \in \{1, \ldots, n\}$. Then C can be written as $\nu b.[b]C'$, and $\models_{\nu c.\nu s.\eta} C'\{c/b\} : \phi_i\{c/a\}$ for all i where c is fresh. Assume that $x:\phi_1, \ldots, x:\phi_n \models^{s,a} x:\psi_1, \ldots, \psi_m$. Then $x:\phi_1\{c/a\}, \ldots, x:\phi_n\{c/a\} \models^{s,c} x:\psi\{c/a\}$ too, since c was fresh. Thus $\models_\eta \nu c.\nu s.C'\{c/b\} : \psi\{c/a\}$, showing that $\models_\eta \nu s.\nu b.[b]C' : \nu a \leftarrow \psi$ as required.

($\to$-MON):

$$\frac{\Gamma, x:\phi_1, \ldots, x:\phi_n \vdash^s x:\psi_1, \ldots, \psi_m}{\Gamma, x:a\to\phi_1, \ldots, x:a\to\phi_n \vdash^s x:a\to\psi_1, \ldots, a\to\psi_m} \quad (a \notin s)$$

($\leftarrow$-MON):

$$\frac{\Gamma, x:\phi_1, \ldots, x:\phi_n \vdash^s x:\psi_1, \ldots, \psi_m}{\Gamma, x:a\leftarrow\phi_1, \ldots, x:a\leftarrow\phi_n \vdash^s x:a\leftarrow\psi_1, \ldots, a\leftarrow\psi_m} \quad (a \notin s)$$

($\nu\to$-MON):

$$\frac{\Gamma, x:\phi_1, \ldots, x:\phi_n \vdash^s x:\psi_1, \ldots, \psi_m}{\Gamma, x:\nu a\to\phi_1, \ldots, x:\nu a\to\phi_n \vdash^s x:\nu a\to\psi_1, \ldots, \nu a\to\psi_m} \quad (a \text{ fresh})$$

($\nu\leftarrow$-MON):

$$\frac{\Gamma, x:\phi_1, \ldots, x:\phi_n \vdash^{s,a} x:\psi_1, \ldots, \psi_m}{\Gamma, x:\nu a\leftarrow\phi_1, \ldots, x:\nu a\leftarrow\phi_n \vdash^s x:\nu a\leftarrow\psi_1, \ldots, \nu a\leftarrow\psi_m} \quad (a \text{ fresh})$$

Table 4: Monotonicity rules for io-modalities

Parallel Composition The rules for parallel composition are shown on table 5.

Input

$$(\to\text{-IN}) \quad \frac{\Gamma, a=b \vdash^s A:\phi}{\Gamma \vdash^s (a)A : b\to\phi} \quad (a \text{ fresh}, b \notin s)$$

$$(\nu\to\text{-IN}) \quad \frac{\Gamma, a\ \mathit{fresh} \vdash^s A:\phi\{a/b\}}{\Gamma \vdash^s (a)A : \nu b\to\phi} \quad (a \text{ fresh})$$

In the context of a rule such as ($\nu\to$-IN) we use $a\ \mathit{fresh}$ as abbreviation of the formula $\wedge\{a \neq c \mid c \text{ not fresh}\}$.

Output

$$(\leftarrow\text{-OUT}) \quad \frac{\Gamma \vdash^s C : a=b \quad \Gamma \vdash^s C:\phi}{\Gamma \vdash^s \langle a\rangle C : b\leftarrow\phi}$$

$$(\nu\leftarrow\text{-OUT}) \quad \frac{\Gamma \vdash^s C:\phi}{\Gamma \vdash^{s,a} \langle a\rangle C : \nu a\leftarrow\phi}$$

Input Modality Conversion Falling a little outside the patterns so far established we also need rule to convert between the free and bound input modalities:

$$(\nu\to\text{-}\to\text{-CONV}) \quad \frac{\cdot}{\Gamma, x:\nu a\to\phi \vdash^{()} x:a\to\phi} \quad (a \notin fn(\Gamma))$$

$$(\to\text{-}\nu\to\text{-CONV}) \quad \frac{\cdot}{\Gamma, x:a\to\phi \vdash^{()} x:\nu a\to\phi} \quad (a \notin fn(\Gamma))$$

($\to$-PAR):

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} F : \mathit{isP} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} E : a \to \phi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} E : a \to \phi_n \\ \Gamma, x : \phi_1, \ldots, x : \phi_n \vdash^{s} x \mid F : \gamma\end{array}}{\Gamma \vdash^{s} E \mid F : a \to \gamma} \quad (a \notin s)$$

($\leftarrow$-PAR):

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} F : \mathit{isP} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} E : a \leftarrow \phi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} E : a \leftarrow \phi_n \\ \Gamma, x : \phi_1, \ldots, x : \phi_n \vdash^{s} x \mid F : \gamma\end{array}}{\Gamma \vdash^{s} E \mid F : a \leftarrow \gamma} \quad (a \notin s)$$

($\nu \to$-PAR):

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} F : \mathit{isP} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} E : \nu a \to \phi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} E : \nu a \to \phi_n \\ \Gamma, x : \phi_1, \ldots, x : \phi_n \vdash^{s} x \mid F : \gamma\end{array}}{\Gamma \vdash^{s} E \mid F : \nu a \to \gamma} \quad (a \text{ fresh})$$

($\nu \leftarrow$-PAR):

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} F : \mathit{isP} \\ \Gamma, s\ \mathit{fresh} \vdash^{()} E : \nu a \leftarrow \phi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} E : \nu a \leftarrow \phi_n \\ \Gamma, x : \phi_1, \ldots, x : \phi_n \vdash^{s} x \mid F : \gamma\end{array}}{\Gamma \vdash^{s} E \mid F : \nu a \leftarrow \gamma} \quad (a \text{ fresh})$$

($\to$-$\leftarrow$-PAR):

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} E : a \to \phi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} E : a \to \phi_n \\ \Gamma, s\ \mathit{fresh} \vdash^{()} F : a \leftarrow \psi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} F : a \leftarrow \psi_m \\ \Gamma, x : \phi_1, \ldots, x : \phi_n, y : \psi_1, \ldots, y : \psi_m \vdash^{s} x \mid y : \gamma\end{array}}{\Gamma \vdash^{s} E \mid F : \gamma}$$

($\nu \to$-$\nu \leftarrow$-PAR):

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} E : \nu a \to \phi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} E : \nu a \to \phi_n \\ \Gamma, s\ \mathit{fresh} \vdash^{()} F : \nu a \leftarrow \psi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} F : \nu a \leftarrow \psi_m \\ \Gamma, x : \phi_1, \ldots, x : \phi_n, y : \psi_1, \ldots, y : \psi_m \vdash^{s,a} x \mid y : \gamma\end{array}}{\Gamma \vdash^{s} A \mid B : \gamma} \quad (a \text{ fresh})$$

Table 5: Rules for parallel composition vs. io-modalities

As an example we show how to derive a rule matching bound output with free input.

Proposition 5.5 *The following rule is derivable:*

$$\frac{\begin{array}{c}\Gamma, s\ \mathit{fresh} \vdash^{()} A : \forall a.a \to \phi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} A : \forall a.a \to \phi_n \\ \Gamma, s\ \mathit{fresh} \vdash^{()} C : \nu a \leftarrow \psi_1, \ldots, \Gamma, s\ \mathit{fresh} \vdash^{()} C : \nu a \leftarrow \psi_m \\ \Gamma, x : \phi_1, \ldots, x : \phi_n, y : \psi_1, \ldots, y : \psi_m \vdash^{s,a} x \mid y : \gamma\end{array}}{\Gamma \vdash^{s} A \mid C : \gamma} \quad (a\ \mathit{fresh})$$

($\to$-ν $\leftarrow$-COM)

PROOF: From the antecedents concerning A first fresh a's are introduced using ($\forall$-R), and then the resulting free inputs are converted to bound inputs using ($\to$-ν $\to$-CONV), so that (ν $\to$-ν $\leftarrow$-COM) can be used to yield the result.

5.8 Examples

In this section we give a first little more substantial proof example. More proof examples are given later.

Example 5.6 Consider the processes

$$\begin{aligned} P &= \overline{a}.\nu b.\langle b\rangle b(c).0 \\ Q &= a(d).\overline{d}\langle d\rangle.0 \end{aligned}$$

In $P \mid Q$ first b is passed as a private name from P to Q along a, then b is returned along itself back to P again. Clearly the judgment

$$\vdash P \mid Q : <\tau><\tau>[\tau]\bot \tag{21}$$

is valid. First CUT-1 is applied. Let

$$\begin{aligned} STOP &= [\tau]\bot \wedge \forall a.[a]\bot \wedge \forall a.[\overline{a}]\bot \\ \phi_1 &= <\overline{a}>\nu b \leftarrow <b>\forall c.c \to STOP \\ \phi_2 &= <a>\forall d.d \to <\overline{d}>d \leftarrow STOP \end{aligned}$$

Using CUT-1 twice we can reduce (21) to the following three subgoals:

$$\vdash P : \phi_1 \tag{22}$$

$$\vdash Q : \phi_2 \tag{23}$$

$$x : \phi_1, y : \phi_2 \vdash x \mid y : <\tau><\tau>[\tau]\bot \tag{24}$$

To prove (22) first apply ($<l>$-ACT) and (REFL) to reduce to

$$\vdash \nu b.\langle b\rangle b(c).0 : \nu b \leftarrow <b>\forall c.c \to STOP \tag{25}$$

and then using (NEW2) and (ν $\leftarrow$-OUT) to

$$\vdash b(c).0 : <b>\forall c.c \to STOP \tag{26}$$

A few more steps in a similar vein suffices to complete the proof of (22), and the proof of (23) is similar. To prove (24) we first reduce to the following list of subgoals

$$x : \phi_1, y : \phi_2 \vdash x : <\overline{a}>\nu b \leftarrow <b>\forall c.c \rightarrow STOP \tag{27}$$

$$x : \phi_1, y : \phi_2 \vdash y : <a>\forall d.d \rightarrow <\overline{d}>d \leftarrow STOP \tag{28}$$

$$x : \phi_1, y : \phi_2, x' : \nu b \leftarrow <b>\forall c.c \rightarrow STOP \vdash x' : isCb \tag{29}$$

$$x : \phi_1, y : \phi_2, y' : \forall d.d \rightarrow <\overline{d}>d \leftarrow STOP \vdash y' : isA \tag{30}$$

$$x : \phi_1, y : \phi_2, x' : \nu b \leftarrow <b>\forall c.c \rightarrow STOP, y' : \forall d.d \rightarrow <\overline{d}>d \leftarrow STOP \vdash x' \mid y' : <\tau>[\tau]\bot \tag{31}$$

using ($<\tau>$-PAR). The first two subgoals are just instances of (I). For (29) we just need to use the monotonicity rule ($\nu \leftarrow$-MON), and for (30) the rules ($\vee$-R), (W-R), ($\forall$-L), ($\forall$-R), and ($\rightarrow$-MON). Finally, for (31), we first use ($\forall$-L) to get

$$x : \cdots, y : \cdots, x' : \nu b \leftarrow <b>\forall c.c \rightarrow STOP, y' : d \rightarrow <\overline{d}>d \leftarrow STOP \vdash x' \mid y' : <\tau>[\tau]\bot \tag{32}$$

where d is fresh, and then, using ($\nu \rightarrow$-$\rightarrow$-CONV) along with (CUT-1), to

$$x : \cdots, y : \cdots, x' : \nu b \leftarrow <b>\forall c.c \rightarrow STOP, y' : \cdots, y'' : \nu d \rightarrow <\overline{d}>d \leftarrow STOP \vdash x' \mid y'' : <\tau>[\tau]\bot \tag{33}$$

which is reduced, by ($\nu \rightarrow$-$\nu \leftarrow$-COM), to

$$x : \cdots, y : \cdots, x' : <b>\forall c.c \rightarrow STOP, y' : \cdots, y'' : <\overline{b}>b \leftarrow STOP \vdash x' \mid y'' : <\tau>[\tau]\bot \tag{34}$$

along with two goals resolved immediately by (I). Proceeding, we use ($<\tau>$-PAR) to reduce to

$$\cdots, x'' : \forall c.c \rightarrow STOP, \cdots, y''' : b \leftarrow STOP \vdash x'' \mid y''' : [\tau]\bot. \tag{35}$$

We now use ($\forall$-L) and ($\rightarrow$-$\leftarrow$-COM) to reduce to

$$\cdots, x''' : STOP, \cdots, y'''' : \ STOP \vdash x''' \mid y'''' : [\tau]\bot. \tag{36}$$

which is resolved very easily using ($[\tau]$-PAR) and some elementary reasoning.

Variable Naming Observe that, because of the shape of the rules, proofs tend to introduce long sequences of variables like x, x', x'', x''' in the example above. It is very often the case that, once transitions or input-output actions are taken (in the form of an application of a modal or an input-output rule), old variables can immediately be forgotten using weakening. In the examples that follow, for this reason we often identify variables like x and x', assuming implicitly that when x' is introduced, assumptions concerning x are immediately forgotten about, whence x' can be renamed to x.

6 Soundness and Completeness for the Modal Fragment

Before pushing on to add rules of discharge that allow interesting recursive properties to be proved we pause to establish soundness and completeness for the modal fragment. Proofs in this section have been deferred to appendix A. Soundness of the most delicate rules was shown as the proof system was presented, so here we can just state soundness as a fact.

Proposition 6.1 (Soundness, Modal Fragment) *If* $\Gamma \vdash^s E : \Delta$ *in the proof system of section 5 then* $\Gamma \models^s E : \Delta$. □

Concerning completeness we consider this only in a rather weak form, namely that if $\Gamma \models^s E : \phi$ where Γ is elementary (i.e. all formulas in Γ are elementary) then $\Gamma \vdash^s E : \phi$ is provable. We first introduce some basic tools.

Definition 6.2 Let η be a partition and N a finite set of names.

1. $hyp(\eta, N)$ denotes a sequence of the form $x_1 : \phi_1, \ldots, x_n : \phi_n$ where each ϕ_i has the form $a = b$ or $a \neq b$ with $a, b \in N$, and where $a = b$ ($a \neq b$) is in $hyp(\eta, N)$ if and only if $\eta \models a = b$ ($\eta \models a \neq b$).

2. Let $\Gamma = x_1 : \psi_1, \ldots, x_n : \psi_n$ where all ψ_i are elementary. Then $\eta \models \Gamma$ if and only if $\eta \models \psi_i$ for all $i : 1 \leq i \leq n$.

We can now show the following property that deals with the first-order part.

Proposition 6.3 *Let N include the set of non-fresh names of* $\Gamma \vdash^s E : \phi$.

1. *If ϕ is boolean then $hyp(\eta, N) \models^s E : \phi$ if and only if $hyp(\eta, N) \vdash^s E : \phi$.*

2. *$\Gamma \vdash^s E : \phi$ if and only if $hyp(\nu s.\eta, N) \vdash^s E : \phi$ for all η such that $\nu s.\eta \models \Gamma$.* □

From this point onwards we shall not be very explicit about the handling of elementary connectives. We prove completeness via a kind of "Decomposition Lemma" allowing us to decompose a goal into subgoals for subagents.

Lemma 6.4 (Decomposition) *Let N include $fn(E\{E_1/x_1, \ldots, E_n/x_n\})$, $fn(s)$, and $fn(\phi)$. Suppose $E\{E_1/x_1, \ldots, E_n/x_n\}$ is closed, and suppose ϕ is non-recursive. Let $\Gamma = hyp(\nu s.\eta, N)$. If $\Gamma \models^s E\{E_1/x_1, \ldots, E_n/x_n\} : \phi$ then there are $\phi_1, \ldots, \phi_n$ of modal depth not exceeding that of ϕ such that*

1. *for all $i : 1 \leq i \leq n$, $\Gamma, s\ fresh \models^{\emptyset} E_i : \phi_i$, and*

2. *$\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \phi$.* □

Theorem 6.5 (Completeness, Modal Fragment) *Suppose that $\Gamma \models^s E : \Delta$, Γ is boolean, and all formulas in Δ are non-recursive. Then $\Gamma \vdash^s E : \Delta$.* □

7 Proof Rules for Recursive Formulas

We now proceed to address fixed points. All approaches to analysis or verification of μ-calculus relies at some level on approximation ordinals and well-founded induction, using the Knaster-Tarski Fixed Point Theorem. So indeed does ours. In some cases when fixed point formulas are unfolded it is possible to determine suitable approximation ordinals which provide progress measures towards satisfaction. This applies, in particular, when unfolding least fixed point formulas to the left of the turnstile, and when unfolding greatest fixed point formulas to the right. Approximation ordinals are reflected explicitly in the proof system, by specific ordinal variables. This provides a simple framework for dealing with a variety of complications including alternation of fixed points and, more importantly in fact, a number of complications related to fixed point interference which we explain below.

The material in this section is based on corresponding material in the paper [6]. For this reason, proofs of some theorems have been left out of this presentation.

Ordinal Approximations Soundness of fixed point induction relies on the well-known iterative characterisation where least and greatest fixed points are "computed" as iterative limits of their ordinal approximations. Let κ range over ordinal variables. Name interpretations are extended to map ordinal variables to ordinals. Let U, V range over fixed point formula abstractions of the form $\sigma X(\vec{a}).\phi$. New formulas are introduced of the shape U^κ and $\kappa < \kappa'$. Ordinal inequalities have their obvious semantics, and $\kappa \leq \kappa'$ abbreviates $\kappa < \kappa' \vee \kappa = \kappa'$ as usual. For approximated fixed point abstractions suppose first that $U = \sigma X(\vec{a}).\phi$ and $\sigma = \nu$. Then

$$\|U^\kappa(\vec{b})\|(\rho,\eta) = \begin{cases} \mathcal{P} \cup \mathcal{A} \cup \mathcal{C}, & (\eta(\kappa) = 0) \\ \|\phi\|(\rho\{\|U^\kappa\|/X\}, \eta\{\eta(\kappa) - 1/\kappa, \vec{b}/\vec{a}\}), & (\eta(\kappa)\ \textit{succ. ord.}) \\ \bigcap\{\|U^\kappa(\vec{b})\|(\rho,\eta') \mid (*)\}, & (\eta(\kappa)\ \textit{limit ord.}) \end{cases}$$

where the condition $(*)$ is that $\eta'(\kappa) < \eta(\kappa)$ and whenever $x \neq \kappa$ then $\eta'(x) = \eta(x)$. Dually, if $\sigma = \mu$:

$$\|U^\kappa(\vec{b})\|(\rho,\eta) = \begin{cases} \emptyset, & (\eta(\kappa) = 0) \\ \|\phi\|(\rho\{\|U^\kappa\|/X\}, \eta\{\eta(\kappa) - 1/\kappa, \vec{b}/\vec{a}\}), & (\eta(\kappa)\ \textit{succ. ord.}) \\ \bigcup\{\|U^\kappa(\vec{b})\|(\rho,\eta') \mid (*)\}, & (\eta(\kappa)\ \textit{limit ord.}) \end{cases}$$

with the same side condition $(*)$ as above. We get the following basic monotonicity properties of ordinal approximations:

Proposition 7.1 *Suppose that $\eta(\kappa) \leq \eta'(\kappa)$ and whenever $x \neq \kappa$ then $\eta'(x) = \eta(x)$.*

1. *If U is a greatest fixed point abstraction then*

$$\|U^\kappa(\vec{b})\|(\rho,\eta') \subseteq \|U^\kappa(\vec{b})\|(\rho,\eta)$$

2. If U is a least fixed point abstraction then

$$\|U^\kappa(\vec{b})\|(\rho,\eta) \subseteq \|U^\kappa(\vec{b})\|(\rho,\eta')$$

PROOF: By wellfounded induction. □

Moreover, and most importantly, we get the following straightforward application of the well-known Knaster-Tarski fixed point theorem.

Theorem 7.2 (Knaster-Tarski) *Suppose that $U = \sigma X(\vec{a}).\phi$. Then*

$$\|U(\vec{b})\|(\rho,\eta) = \begin{cases} \cap\{\|U^\kappa(\vec{b})\|(\rho,\eta\{\alpha/\kappa\}) \mid \alpha \text{ an ordinal}\}, & \text{if } \sigma = \nu \\ \cup\{\|U^\kappa(\vec{b})\|(\rho,\eta\{\alpha/\kappa\}) \mid \alpha \text{ an ordinal}\}, & \text{if } \sigma = \mu \end{cases}$$

As the intended model is countable the quantification in theorem 7.2 can be restricted to countable ordinals.

7.1 Rules for Fixed Point Unfolding and Approximation

The main rules to reason locally about fixed point formulas are the unfolding rules. These come in four flavours, according to whether the fixed point abstraction concerned has already been approximated or not, and to the nature and position of the fixed point relative to the turnstile.

$$(\text{APPRX} - \text{L}) \quad \frac{\Gamma, x : U^\kappa(\vec{b}) \vdash^s E : \Delta}{\Gamma, x : U(\vec{b}) \vdash^s E : \Delta} U \text{ lfp, } \kappa \text{ fresh}$$

$$(\text{APPRX} - \text{R}) \quad \frac{\Gamma \vdash^s E : U^\kappa(\vec{b}), \Delta}{\Gamma \vdash^s E : U(\vec{b}), \Delta} U \text{ gfp, } \kappa \text{ fresh}$$

$$(\text{UNF} - \text{L} - 1) \quad \frac{\Gamma, x : \phi\{U/X, \vec{b}/\vec{a}\} \vdash^s E : \Delta}{\Gamma, x : U(\vec{b}) \vdash^s E : \Delta} U = \sigma X(\vec{a}).\phi$$

$$(\text{UNF} - \text{R} - 1) \quad \frac{\Gamma \vdash^s E : \phi\{U/X, \vec{b}/\vec{a}\}, \Delta}{\Gamma \vdash^s E : U(\vec{b}), \Delta} U = \sigma X(\vec{a}).\phi$$

$$(\text{UNF} - \text{L} - 2) \quad \frac{\Gamma, x : \phi\{U^{\kappa_1}/X, \vec{b}/\vec{a}\}, \kappa_1 < \kappa \vdash^s E : \Delta}{\Gamma, x : U^\kappa(\vec{b}) \vdash^s E : \Delta} \quad U = \mu X(\vec{a}).\phi, \kappa_1 \text{ fresh}$$

$$(\text{UNF} - \text{R} - 2) \quad \frac{\Gamma, \kappa_1 < \kappa \vdash^s E : \phi\{U^{\kappa_1}/X, \vec{b}/\vec{a}\}, \Delta}{\Gamma \vdash^s E : U^\kappa(\vec{b}), \Delta} \quad U = \nu X(\vec{a}).\phi, \kappa_1 \text{ fresh}$$

$$(\text{UNF} - \text{L} - 3) \quad \frac{\Gamma, x : \kappa_1 < \kappa \supset \phi\{U^{\kappa_1}/X, \vec{b}/\vec{a}\} \vdash^s E : \Delta}{\Gamma, x : U^\kappa(\vec{b}) \vdash^s E : \Delta} \quad U = \nu X(\vec{a}).\phi, \kappa_1 \text{ fresh}$$

$$(\text{UNF} - \text{R} - 3) \quad \frac{\Gamma \vdash^s E : \kappa_1 < \kappa \wedge \phi\{U^{\kappa_1}/X, \vec{b}/\vec{a}\}, \Delta}{\Gamma \vdash^s E : U^\kappa(\vec{b}), \Delta} \quad U = \mu X(\vec{a}).\phi, \kappa_1 \text{ fresh}$$

The first unfolding rules, $(\text{UNF} - \text{L} - 1)$ and $(\text{UNF} - \text{R} - 1)$, are the expected unfolding rules. These rules are always used when unfolding least fixed point formulas occurring to the right of the turnstile, or, dually, greatest fixed point

formulas occurring to the left. In these cases the proof task is an existential one, to identify some approximation ordinal making the statement true, which the first unfolding rules merely serve to delay. On the other hand, in the case of least fixed point formulas occurring to the left of the turnstile, or greatest fixed point formulas occurring to the right, the task is a universal one, suggesting well-founded induction as a suitable proof strategy. The approximation rules, (Apprx − l) and (Apprx − r), serve to introduce ordinal variables for this purpose. Having introduced ordinal variables they need to decremented as approximated formulas are unfolded. This is the purpose of the second pair of unfolding rules, (Unf − l − 2) and (Unf − r − 2).

Now, since ordinal approximations are introduced at only certain positions in a judgment (to the left for least fixed points and to the right for greatest ones), if the positions of approximated formulas would be unaffected by the local proof rules, the six rules so far discussed would have been sufficient. Unfortunately, due to the cut rules, this is not so. Consider for instance the following (quite typical) application of the process cut rule:

$$\frac{\Gamma \vdash^{()} Q : U^{\kappa} \quad \Gamma, x : U^{\kappa} \vdash^{()} P : U^{\kappa}}{\Gamma \vdash^{()} P\{Q/x\} : U^{\kappa}}$$

In this example U may be a greatest fixed point formula which, through some earlier application of (Apprx-r) has been assigned the ordinal variable κ. The second antecedent has U^{κ} occurring to the left of the turnstile. The third pair of unfolding rules are needed to handle this situation.

In addition to the above 8 rules it is useful also to add versions of the identity rules reflecting the monotonicity properties of ordinal approximations, prop. 7.1:

$$\mathsf{IdMon1} \quad \frac{\Gamma \vdash \kappa \leq \kappa'}{\Gamma, x : U^{\kappa}(\vec{b}) \vdash^{()} x : U^{\kappa'}(\vec{b}), \Delta} \quad U \text{ lfp}$$

$$\mathsf{IdMon2} \quad \frac{\Gamma \vdash \kappa \geq \kappa'}{\Gamma, x : U^{\kappa}(\vec{b}) \vdash^{()} x : U^{\kappa'}(\vec{b}), \Delta} \quad U \text{ gfp}$$

Additionally a set of elementary rules are needed to support reasoning about well-orderings, including transitivity and irreflexivity of $<$. These rules are left out of the presentation. For the soundness proof (which is uncontroversial) we refer to [6].

Theorem 7.3 *The rules* (Apprx-l), (Apprx-r), (Unf-l-1), (Unf-r-1), (Unf-l-2), (Unf-r-2), (Unf-l-3), (Unf-r-3), IdMon1, *and* IdMon2 *are sound.* □

7.2 Rule of Discharge

In addition to the local rules for unfolding and approximating fixed point formulas, a rule is needed for discharging valid induction hypothesis instances. The fundamental problem in devising such a rule is that fixed points may interfere as proofs are elaborated. The problem is illustrated in figure 2. The formula U is assumed to be a least fixed point formula, and the formula V is a greatest fixed point formula. The node labelled $*$ can be interpreted locally

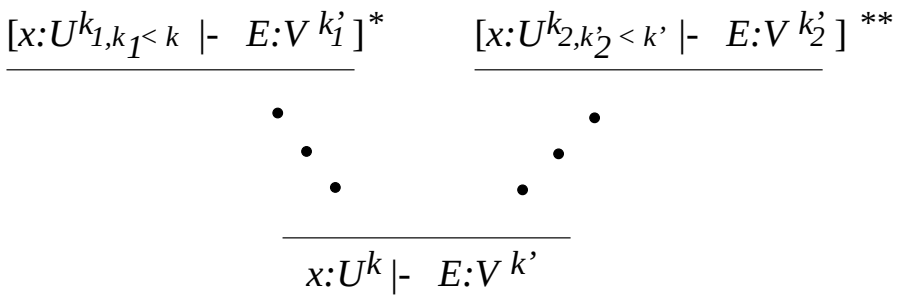

Figure 2: Fixed point interference

as an instance of an induction hypothesis, using induction on κ, and the node labelled $**$ similarly uses induction on κ'. However, for the node $*$ there is no information as to the relationship between κ'_1 and κ', and similarly for the node $**$ there is nothing relating κ'_2 and κ'. This easily happens in practice, viz. the examples below. The problem is that in this case the unfolding of fixed points interfere: With the information provided we are unable to cast the proof as a proof by well-founded induction with the nodes $*$ and $**$ corresponding to applications of an inductive hypothesis for the simple reason that such an argument would be unsound. On the other hand, if we knew at node $**$, say, that $\kappa_2 \leq \kappa$, such a casting *would* exist, as nested induction first on κ and then in κ'.

That this problem indeed arises in practice is illustrated by the following two examples. The first example shows where discharge should fail because of fixed point interference.

Example 7.4 Consider the proof goal

$$x : \nu Z_1.\mu Z_2.[\tau]Z_1 \wedge [a]Z_2 \vdash^{()} x : \mu Z_3.\nu Z_4.[\tau]Z_4 \wedge \forall a.[a]Z_3 \qquad (37)$$

The assumption states (in the absence of name passing which is not needed to illustrate the problems) that any infinite sequence of transitions labelled τ or a can only contain a finite number of consecutive transitions labelled a, while the conclusion states that any infinite sequence of transitions labelled τ or a can only contain a finite number of a transitions, never mind if consecutive or not. Thus (37) is false. We attempt to build a proof for (37) to see where the construction breaks down.

Let us introduce the following abbreviations:

$$\begin{aligned} U_1 &= \nu Z_1.\mu Z_2.[\tau]Z_1 \wedge [a]Z_2 \\ U_2 &= \mu Z_2.[\tau]U_1 \wedge [a]Z_2 \\ U_3 &= \mu Z_3.\nu Z_4.[\tau]Z_4 \wedge [a]Z_3 \\ U_4 &= \nu Z_4.[\tau]Z_4 \wedge [a]U_3 \end{aligned}$$

We start by refining (37) to the subgoal

$$x : U_2^{\kappa_2} \vdash^{()} x : U_4^{\kappa_4} \tag{38}$$

using the rules (Unf-l-1), (Unf-r-1), (Apprx-l) and (Apprx-r). Continuing a few steps further (by unfolding the fixed point formulas and treating the conjunctions on the left and on the right) we obtain the two subgoals

$$x : [\tau]U_1, x : [a]U_2^{\kappa_2'}, \kappa_2' < \kappa_2, \kappa_4' < \kappa_4 \vdash^{()} x : [\tau]U_4^{\kappa_4'} \tag{39}$$

$$x : [\tau]U_1, x : [a]U_2^{\kappa_2'}, \kappa_2' < \kappa_2, \kappa_4' < \kappa_4 \vdash^{()} x : [a]U_3 \tag{40}$$

Subgoal (39) is refined via rule Mon2 to

$$x' : U_1, x : [a]U_2^{\kappa_2'}, \kappa_2' < \kappa_2, \kappa_4' < \kappa_4 \vdash^{()} x' : U_4^{\kappa_4'} \tag{41}$$

and after unfolding U_1 using (Unf-l-1) and then approximating we arrive at

$$x' : U_2^{\kappa_2''}, x : [a]U_2^{\kappa_2'}, \kappa_2' < \kappa_2, \kappa_4' < \kappa_4 \vdash^{()} x' : U_4^{\kappa_4'}. \tag{42}$$

This judgment we might hope to be able to discharge against (38) by induction on κ_4. By the same token when we refine (40) to

$$x : [\tau]U_1, x' : U_2^{\kappa_2'}, \kappa_2' < \kappa_2, \kappa_4' < \kappa_4 \vdash^{()} x' : U_4^{\kappa_4''} \tag{43}$$

we would expect to be able to discharge against (38) inductively on κ_2. This does not work, however, since derivation of (42) from (38) fails to preserve the induction variable κ_2 needed for (43), and vice versa, κ_4 is not preserved along the path from (38) to (43).

Secondly we give an example showing where discharge should succeed.

Example 7.5 Consider the (reversed) proof goal

$$x : \mu Z_1.\nu Z_2.[\tau]Z_2 \wedge [a]Z_1 \vdash^{()} x : \nu Z_3.\mu Z_4.[\tau]Z_3 \wedge [a]Z_4 \tag{44}$$

stating that if all infinite sequences of transitions labelled τ or a can only contain a finite number of a transitions, then these infinite sequences of τ or a transitions can only contain finite sequences of *consecutive* a transitions. This goal is clearly valid.
The abbreviations we shall use are:

$$\begin{aligned} U_1 &= \mu Z_1.\nu Z_2.[\tau]Z_2 \wedge [a]Z_1 \\ U_2 &= \nu Z_2.[\tau]Z_2 \wedge [a]U_1^{\kappa_1'} \\ U_3 &= \nu Z_3.\mu Z_4.[\tau]Z_3 \wedge [a]Z_4 \\ U_4 &= \mu Z_4.[\tau]U_3^{\kappa_3'} \wedge [a]Z_4 \end{aligned}$$

First we apply rules (Apprx-l), (Apprx-r), (Unf-l-2) and (Unf-r-2) to reduce (44) to the subgoal

$$x : U_2, \kappa_1' < \kappa_1, \kappa_3' < \kappa_3 \vdash^{()} x : U_4 \tag{45}$$

Continuing in much the same way as in the preceding example we arrive at the two subgoals

$$x' : U_2, x : [a]U_1^{\kappa_1'}, \kappa_1' < \kappa_1, \kappa_3' < \kappa_3 \vdash^{()} x' : U_3^{\kappa_3'} \quad (46)$$

$$x : [\tau]U_2, x' : U_1^{\kappa_1'}, \kappa_1' < \kappa_1, \kappa_3' < \kappa_3 \vdash^{()} x' : U_4 \quad (47)$$

These subgoals are refined, using (UNF-R-2) and (UNF-L-2) respectively, to

$$x' : U_2, x : [a]U_1^{\kappa_1'}, \kappa_1' < \kappa_1, \kappa_3' < \kappa_3, \kappa_3'' < \kappa_3' \vdash^{()} x' : \mu Z_4.[\tau]U_3^{\kappa_3''} \wedge [a]Z_4 \quad (48)$$

$$x : [\tau]U_2, x' : \nu Z_2.[\tau]Z_2 \wedge [a]U_1^{\kappa_1''}, \kappa_1' < \kappa_1, \kappa_3' < \kappa_3, \kappa_1'' < \kappa_1' \vdash^{()} x' : U_4 \quad (49)$$

In this case it is safe to discharge both judgments against (45), since the unfolding of U_2 does not interfere with that of U_4.

The Rule of Discharge There are two key task in providing a sound and generally applicable rule of discharge:

- The first task is to ensure that each discharged node determines a "progressing cycle" in the proof structure: That is, that the node determines a cycle, and that along that cycle some ordinal variable is decreased in a recursive manner.
- The second task is to ensure that progress required for the safe discharge of one node can not be undone by cycles induced by the discharge of some other node. Or in other words, in traversing cycles induced by node discharge critical progress ordinals belonging to other cycles must be *regenerated*.

It would be possible to use games to completely characterize the conditions under which discharge is safe. Such a game-based characterisation, however, would be too global a condition to be very useful for proof construction (which is the main aim of the work reported here), and it is often helpful to trade a complete, but global, rule for an incomplete one which is more local. Other versions of a rule of discharge than the one presented here can be devised, cf. [7] for an example applied to CCS. The present rule of discharge is based on [6].

The rule of discharge relies on some fixed, but arbitrary linear ordering $<$ on fixed point formula abstractions U. Assuming such a single fixed linear ordering can be too restrictive when recursive proof structures are independent. For the purpose of the examples and theorems of the rest of the paper this is, however, not a problem. Below we briefly discuss ways of relaxing the construction to allow the linear ordering to be built incrementally.

Below we define the critical notions of regeneration, progress, and discharge. We use v to range over proof structure nodes. Discharge is applied when facing a proof goal v_n which is unelaborated (no rule has been applied to that node), such that, below v_n we find some already elaborated node v_1 such that v_n is in a sense an instance of v_1. This requires names and term variables present in v_1 to be interpreted as names in v_n. This is what the substitution η of the following definition serves to achieve.

Definition 7.6 (Regeneration, Progress, Discharge) Let $\Pi = v_1, \ldots, v_n$ be a path such that v_n is not elaborated. Suppose that v_i is labelled by $\Gamma_i \vdash^{s_i} E_i : \Delta_i$ for all $i : 1 \leq i \leq n$.

1. The path Π is *regenerative* for U and the name interpretation η, if whenever there is a κ_i such that U^{κ_i} is a subformula of Γ_i (Δ_i) then there are also $\kappa_1, \ldots, \kappa_{i-1}, \kappa_{i+1}, \ldots, \kappa_n$ such that for all $j : 1 < j \leq n$, U^{κ_j} is a subformula of Γ_j (Δ_j), and $\Gamma_j \vdash^{()} \kappa_j \leq \kappa_{j-1}$. Moreover we require that $\eta(\kappa_1) = \kappa_n$.

2. The path Π is *progressive* for U and η if we can find $\kappa_1, \ldots, \kappa_n$ such that:

   (a) For all $i : 1 < i \leq n$, U^{κ_i} is a subformula of Γ_i (Δ_i), and $\Gamma_i \vdash^{()} \kappa_i \leq \kappa_{i-1}$.

   (b) $\eta(\kappa_1) = \kappa_n$.

   (c) For some $i : 1 < i \leq n$, $\Gamma_i \vdash^{()} \kappa_i < \kappa_{i-1}$.

3. The node v_n can be *discharged* against the node v_1 if we can find some U and substitution η such that:

   (a) Π is regenerative for all $U' < U$ and η.

   (b) Π is progressive for U and η.

   (c) $E_n = E_1\eta$ and $s_n = s_1\eta$.

   (d) For all assumptions $x : \phi$ in Γ_1, $\Gamma_n \vdash^{()} x\eta : \phi\eta$, and all assertions ϕ in Δ_1 then $y : \phi\eta \vdash^{()} y : \Delta_n$.

   In this case we term v_n a *discharge node* and v_1 its *companion node.*

In this definition we are being slightly sloppy with our use of U's: Really we are identifying fixed point formula abstractions up to ordinal approximations except where they are explicitly stated.

Condition 7.6 first states that discharge can take place on an unelaborated node if we find some ancestral node which is "more general" (condition 7.6.3.c and d) than the node v_n being discharged, provided that the cycle thus induced satisfies the regeneration and progress constraints given. Observe that an efficient approximation of condition 7.6.3.d would be to test for membership, ie. $s\eta : \phi\eta \in \Gamma_n$ and $\phi\eta \in \Delta_n$ respectively. With this condition the name interpretation η becomes extendable to a substitution making $\Gamma_1 \vdash^{s_1} E_1 : \Delta_1$ identical to a weakened version of $\Gamma_n \vdash^{s_n} E_n : \Delta_n$.

There does not appear to be any obvious way in which the equality constraint 7.6.3.c on restriction sets can be eased. Instead the second cut rule (CUT-2) must be used to explicitly garbage-collect unused names as proof elaboration proceeds.

The progress condition, 7.6.2, requires the existence of a cycle on an ordinal variable such that, along the cycle, the value associated to that ordinal variable is somewhere strictly decreased.

Finally, the regeneration condition, 7.6.1, is the condition required to ensure that progress cycles on formulas V with "higher priority" (smaller in the

ordering $<$) than the formula U currently being considered can not be undone when the cycles are nested.

Concerning the examples, it is quite easy to verify that for Example 7.4 no linearisation of the fixed point formulas can be devised such that the nodes (41) and (42) can be discharged. On the other hand, for Example 7.5, any linear ordering which (up to approximation ordinals) has $U_4 < U_2$ will do.

Observe that the linear ordering on fixed point formula abstractions can be chosen quite freely. One might expect some correlation between position in the linear ordering and depth of alternation, viz. example 7.5 above. In practice this is in fact a good guide to choosing a suitable linear ordering. However, as we show, we do not need to require such a correlation a priori. Moreover one can construct examples, using cut's, of proofs for which the above rule of thumb does not work.

Now, the *compositional proof system* is obtained by adding the proof rules for fixed points, including the rule of discharge, to the local rules of section 5. We write $\Gamma \vdash^s_C E : \Delta$ if the judgment $\Gamma \vdash^s E : \Delta$ is provable in the compositional proof system. For the proof of soundness of this proof system we refer the reader to [6],

Theorem 7.7 (Soundness, Compositional Proof System) *If* $\Gamma \vdash^s_C E : \Delta$ *then* $\Gamma \models^s E : \Delta$. □

8 Finite Control Completeness

We then turn to the issue of completeness and consider generalisations of the completeness result (Theorem 6.5) to recursive formulas. The generalisation of Theorem 6.5 which we seek is to general formulas and finite control processes.

Definition 8.1 (Finite Control Agent) Say that an agent term E *uses* the agent identifier D, if either D occurs in E or else D occurs in the body of an identifier D' such that E uses D'. A *finite control agent* is a closed agent term E such that the parallel operator $|$ does not occur in the body of any identifier used by E.

The notion of finite control process is a direct generalisation of the notion of finite state process in the case of CCS. Completeness for finite state (CCS) processes vs a compositional proof system related to the proof system considered here was proved in [5]. However, the details of that formalisation, in particular the rule of discharge, was quite different from the proof system presented here.

To prove completeness for finite control processes we formulate a *model checker*, an alternative, non-compositional proof system and show that whichever judgment is provable using the model checker is also provable in the proof system presented in the preceding sections. Model checkers based on non-compositional proof systems such as the one we go on to present have been considered and proved complete for finite control processes several times over [4, 1]. Similar techniques can be used to prove completeness of the present version.

Definition 8.2 (Model checking judgments) A *model checking judgment* is a judgment of the form $\Gamma \vdash E : \phi$ for which all formulas in Γ are elementary and for which E is of finite control.

We define a proof system to apply to model checking judgments. The proof system consists of all rules defining the compositional proof system (restricted to model checking judgments) excluding the transition and io modality rules. That is, the proof system includes the structural rules (which due to the restriction to model checking judgments become rather standard, for instance the first two cut rules become superfluous), the logical rules, and the rules equality/inequality.

The transition rules are replaced by the following two schemes:

(Poss) If for all η such that $\eta \models \Gamma$ there is an E such that $P\eta \stackrel{l_\tau \eta}{\rightarrow} E$ and $\Gamma \vdash E : \phi$ has been inferred, infer then $\Gamma \vdash P : <l_\tau>\phi$

(Nec) If $\Gamma \vdash E : \phi$ has been inferred for all E such that $P\eta \stackrel{l_\tau \eta}{\rightarrow} E$ and $\eta \models \Gamma$, infer then $\Gamma \vdash P : [l_\tau]\phi$

together with the rules ($\rightarrow$-in), ($\nu \rightarrow$-in), ($\leftarrow$-out), and

$$(\nu \leftarrow\text{-out'}) \quad \frac{\Gamma, a \textit{ fresh} \vdash C : \phi\{a/b\}}{\Gamma \vdash \nu a.\langle a\rangle C : \nu b \leftarrow \phi} \quad (a \text{ fresh})$$

The rule of discharge applies to the model checker without modification. In effect the conditions in this case degenerate to those of the model checker presented in [4]. Observe that the rules (Poss) and (Nec) are infinitary due to the quantification over name interpretations. Name interpretations are only significant, however, up to the names that are not fresh (of which there is only a finite supply). Write $\Gamma \vdash_{MC} E : \phi$ if there is a model checker proof of $\Gamma \vdash E : \phi$.

We state soundness and completeness of the model checker proof system without proof:

Theorem 8.3 (Soundness and Completeness of Model Checker) *For all model checking judgments* $\Gamma \vdash E : \phi$, $\Gamma \vdash_{MC} E : \phi$ *if and only if* $\Gamma \models E : \phi$. $\square$

One main obstacle in proving the completeness result we seek is that we need to devise a strategy for choosing cut-formulas in order to apply the dynamic rules in the compositional proof system. There are many ways of doing this. In practice one normally takes a lazy approach and just introduces a logical variable to stand for the required cut formula, and then gradually instantiate this variable as the need arises. In fact such a strategy may be quite efficient and it is moreover well suited for parallel or distributed implementations. Here, however, we choose a non-lazy approach instead, mainly because it makes the proof easier. Because processes are assumed not to contain | in recursively defined contexts in turns out that such processes can be characterised completely up to (in this case strong late) bisimulation equivalence by a so-called characteristic formula. We have already shown this to be the case for a related π-calculus logic in earlier work [1]. Assume that P is a finite control process. Assume for each well-formed process term Q and name interpretation η a unique formula

variable $X_{Q,\eta}$. Let vis range over sequences of pairs of process terms and name interpretations (those pairs that have already been visited once in constructing the characteristic formula). The formula $Char(P, \eta, vis)$ is the characteristic formula for P, given name interpretation η and visited list vis:

$$Char(E, \eta, vis) = \begin{cases} X_{E,\eta} & \text{if } (E, \eta) \in vis \\ \nu X_{E,\eta}. \bigwedge_{l_\tau} \phi(<l_\tau>) \wedge \bigwedge_{l_\tau} \phi([l_\tau]) & \text{otherwise} \end{cases}$$

Here, $\phi(<l_\tau>)$ describes the potential transitions possible from P assuming η, and $\phi([l_\tau])$ describes the property "necessarily" holding in the continuation:

$$\begin{aligned} \phi(<l_\tau>) &= \bigwedge\{<l_\tau> Char(E, \eta, (P, \eta) :: vis) \mid P\eta \stackrel{l_\tau \eta}{\rightarrow} E\} \\ \phi([l_\tau]) &= [l_\tau] \bigvee\{ Char(E, \eta, (P, \eta) :: vis) \mid P\eta \stackrel{l_\tau \eta}{\rightarrow} E\} \end{aligned}$$

For abstraction and concretions we define:

$$\begin{aligned} Char((a)A, \eta, vis) &= \forall a.a \rightarrow \bigvee\{a = b \wedge Char(A\{b/a\}, \eta, vis) \mid b \in fn((a)A)\} \vee \\ &\quad (\bigwedge\{a \neq b \mid b \in fn((a)A)\} \wedge Char(A, \nu a.\eta, vis)) \\ Char(\langle a \rangle C, \eta, vis) &= a \leftarrow Char(C, \eta, vis) \\ Char(\nu a.\langle a \rangle C, \eta, vis) &= \nu a \leftarrow Char(C, \nu a.\eta, vis) \end{aligned}$$

Abbreviate $Char(E, \eta, \varepsilon)$ by $Char(E, \eta)$.

Observe that only finite conjunctions and disjunctions are used in the definition of *Char*. Thus, the only reasons why $Char(E, \eta, vis)$ could be ill-defined would be if the computation of $Char(E, \eta, vis)$ failed to terminate, and this could easily happen if care is not taken when choosing names of bound variables. However, if we adopt the *conventions* that

1. names are linearly ordered, and every time a variable is bound it is chosen to be minimal in the ordering, and
2. $\nu a.E$ is identified with E whenever a does not appear freely in E,

then it can be shown that $Char(E, \eta, vis)$ is in fact well-defined using the techniques of e.g. [4] or [1]. Here we just state this as a fact.

Proposition 8.4 *Suppose that P is a finite control process. Then $Char(P, \eta)$ is well-defined.* □

For the completeness proof we need to resort to the following correctness property for characteristic formulas. We leave out the proof of this quite easy lemma. A similar result was proved in [1]. In the statement of this lemma and for the remainder of this section we abbreviate $hyp(\eta, N)$ by $hyp(\eta)$ where the set N is any set of names including those free in the judgment under consideration.

Lemma 8.5 *For all finite control processes P and partitions η,*

$$hyp(\eta) \vdash_{MC} P : Char(P, \eta)$$

. □

Before embarking on the main completeness result we need one further lemma, used to pass from results concerning terms with substitutions to results concerning open terms governed by assumptions.

Lemma 8.6 *For all finite control processes P and partitions η, if $hyp(\eta) \vdash_{MC} P : \phi$ then $hyp(\eta), x : Char(P, \eta) \vdash_C x : \phi$.*

PROOF: (Hint) The compositional proof mimicks the model checker proof by appealing to the monotonicity rules. □

The proof of the finite control completeness theorem is deferred to appendix B.

Theorem 8.7 *If $\Gamma \vdash_{MC} E : \phi$ then $\Gamma \vdash_C^{()} E : \phi$.* □

In view of theorem 8.3, completeness for finite control processes is then a direct corollary:

Corollary 8.8 (Finite Control Completeness) *For all model checking judgments $\Gamma \vdash E : \Delta$, if $\Gamma \models E : \Delta$ then $\Gamma \vdash_C^{()} E : \Delta$.* □

9 Natural Numbers

In this section we consider the specification of Nat given in section 4, and show how one can use the proof systems to formally demonstrate that the operations (of $ZERO$, $SUCC$, $COPY$, and ADD) satisfy the desired properties. Observe that while $ZERO$ and $SUCC$ are very simple static processes, $COPY$ and ADD are not.

Proposition 9.1 *The following judgments are derivable:*

1. $\vdash_C^{()} ZERO(n) : Nat(n)$
2. $x : Nat(n) \vdash_C^{n} SUCC(n, m) \mid x : Nat(m)$
3. $x : Nat(n) \vdash_C^{n} COPY(n, m) \mid x : Nat(m)$
4. $x : Nat(n_1), y : Nat(n_2) \vdash_C^{n_1, n_2} ADD(n_1, n_2, m) \mid x \mid y : Nat(m)$ □

We concentrate in this section on giving an outline proof of (3), proving (2) along the way. The proof of (1) is quite straightforward, and the proof of (4) is a variation on the theme.

The main problem which the proof (of (3)) has to face is process creation in the definition of $COPY$. That is, the continuation of $COPY(n, m)$ contains a term of the shape $\nu m_1.(SUCC(m_1, m) \mid COPY(n_1, m_1))$. To deal with this we use a process cut, replacing each of the subprocesses $SUCC(m_1, m)$ and $COPY(n_1, m_1)$ by abstract state-oriented descriptions, and continue proving correctness based on those.

Finding an appropriate cut formula for $SUCC$ is easy:

$$\begin{aligned}
Succ_0(n,m,\phi) &= <m>\top \wedge \\
&\quad [\tau]\bot \wedge \\
&\quad \forall b.[b](b = m \wedge \nu s \rightarrow \nu z \rightarrow \phi(n,s)) \wedge \\
&\quad \forall b.[\overline{b}]\bot \\
Succ_1(n,s) &= <\overline{s}>\top \wedge \\
&\quad [\tau]\bot \wedge \\
&\quad \forall b.[b]\bot \wedge \\
&\quad \forall b.[\overline{b}](b = s \wedge n \leftarrow STOP \\
Succ(n,m) &= Succ_0(n,m,Succ_1)
\end{aligned}$$

The correctness of $Succ$ is easily proved:

Proposition 9.2

1. $\vdash^{()}_C SUCC(n,m) : Succ(n,m)$
2. $x : Nat(n), y : Succ(n,m) \vdash^{n}_C y \mid x : Nat(m)$ □

Observe that this establishes proposition 9.1.2 by means of a process cut.

The next step is to find a cut formula for $COPY$. Our intention is to define a property characterising, to a sufficiently precise degree, the behaviour of a process term of the shape

$$\nu m_1.(SUCC(m_1,m) \mid \nu m_2.SUCC(m_2,m_1) \mid \cdots \mid COPY(n,m_k)\cdots) \qquad (50)$$

Again we use a state machine-oriented style of specification. The machine being predicated will, as (50) have two components, each with a two-state behaviour, namely a component corresponding to the successor processes, and a component corresponding to the copy process. Each of these component can to a large extent execute concurrently. We thus propose the following definition of the cut formula Gcopy (generalized copy) as a greatest fixed point formula:

$$\begin{aligned}
Gcopy_{0,0}(n,m) &= <\overline{n}>\top \wedge [\tau]\bot \wedge \\
&\quad \forall b.[b](b = m \wedge \nu s \rightarrow \nu z \rightarrow Gcopy_{1,0}(n,s,z)) \wedge \\
&\quad \forall b.[\overline{b}](b = n \wedge \nu s_1 \leftarrow \nu z_1 \leftarrow Gcopy_{0,1}(m,s_1,z_1)) \\
Gcopy_{1,0}(n,s,z) &= <\overline{s}>\top \wedge <\overline{n}>\top \wedge [\tau]\bot \wedge \\
&\quad \forall b.[b]\bot \wedge \\
&\quad \forall b.[\overline{b}]((b = s \wedge \nu m_1 \leftarrow Gcopy_{0,0}(n,m_1)) \vee \\
&\qquad (b = n \wedge \nu s_1 \leftarrow \nu z_1 \leftarrow Gcopy_{1,1}(s,z,s_1,z_1))) \\
Gcopy_{0,1}(m,s_1,z_1) &= <z_1>\top \wedge <s_1>\top \wedge [\tau]\bot \wedge \\
&\quad \forall b.[b]((b = z_1 \wedge Nat(m)) \vee \\
&\qquad (b = s_1 \wedge \nu n_1 \rightarrow Gcopy_{0,0}(n_1,m)) \vee
\end{aligned}$$

$$
\begin{array}{rcl}
 & & (b = m \wedge \nu s \rightarrow \nu z \rightarrow Gcopy_{1,1}(s, z, s_1, z_1))) \\
Gcopy_{1,1}(s, z, s_1, z_1) & = & <\overline{s}>\top \wedge <z_1>\top \wedge <s_1>\top \wedge [\tau]\bot \wedge \\
 & & \forall b.[b]((b = z_1 \wedge Nat_1(Nat)(s, z)) \vee \\
 & & \quad (b = s_1 \wedge \nu n_1 \rightarrow Gcopy_{1,0}(n_1, s, z))) \wedge \\
 & & \forall b.[\overline{b}](b = s \wedge \nu m_1 \leftarrow Gcopy_{0,1}(m_1, s_1, z_1))
\end{array}
$$

The definition of *Gcopy* is given in equational terms. However, the definition is easily rewritten as a proper greatest fixed point formula. The correctness of *Gcopy* is reflected by the following proposition:

Proposition 9.3

1. $x : Succ(m_1, m), y : Gcopy_{0,0}(n, m_1) \vdash_C^{m_1} x \mid y : Gcopy_{0,0}(n, m)$
2. $\vdash_C^{()} COPY(n, m) : Gcopy_{0,0}(n, m)$
3. $x : Nat(n), y : Gcopy_{0,0}(n, m) \vdash^n y \mid x : Nat(m)$ □

The formal proof of proposition 9.3 is quite sizable though (since we have identified the cut formulas) routine and likely to be mechanizable. Observe that the proof of 9.3.2 uses 9.3.1 through two process cuts. Finally proposition 9.1.3 is a direct corollary of proposition 9.3.2 and 9.3.3 using a process cut.

It may be worthwhile to make clear that we do not intend to advocate the use π-calculus or π-μ-calculus representations of data types in actual practice. There are much simpler accounts of data types around useful for practical programming and specification. What the example serves to illustrate, however, is that while fairly trivial as a data type, *as processes executing in parallel*, π-calculus representations of data type embody a simple sort of mobility protocol the behaviour and correctness of which is not trivial. Observe in this connection also that we achieve genuinely more than the corresponding type correctness results of [10] which are, in effect, results in the meta theory of the π-calculus whereas here the reasoning has been "internalised" using the π-μ-calculus proof system.

10 Buffers

In this section we consider buffer properties in the style of example 4.3. Consider for instance the formula $NoSpuOut(i, o)$ describing absence of spurious output. Our task in this section is to show that $NoSpuOut(i, o)$ holds of the unbounded garbage-collecting buffer $GCBuf(i, o)$. As usual, since $GCBuf(i, o)$ creates processes dynamically, we use process cut's using cut-formulas which reflect a state machine-like behaviour. First, for start cells define:

$$
\begin{array}{rcl}
Sc(i, o, n, a) & = & [\tau]Sc(i, o, n, a) \wedge \\
 & & \forall b.[b]((b = i \wedge \forall d.d \rightarrow d = a \vee Sc(i, o, n, a)) \vee \\
 & & \quad (b = n \wedge \forall o', n'.i = o' \vee i = n' \vee \\
 & & \quad\quad o' \rightarrow n' \rightarrow Sc(i, o', n', a))) \wedge \\
 & & \forall b.[\overline{b}](b = o \wedge \exists d.d \leftarrow d \neq a \wedge Sc(i, o, n, a))
\end{array}
$$

and then for buffer cells define

$$\begin{aligned} Bc(o,d,n_l,n_r) &= [\tau]\bot \wedge \\ &\quad \forall b.[b](b = n_r \wedge \forall o', n'.i = o' \vee i = n' \vee \\ &\qquad\qquad o' \rightarrow n' \rightarrow Bc(o',d,n_l,n')) \wedge \\ &\quad \forall b.[\overline{b}](b = o \wedge d \leftarrow Bc_1(o,n_l,n_r)) \\ Bc_1(o,n_l,n_r) &= [\tau]\bot \wedge \\ &\quad \forall b.[b]\bot \wedge \\ &\quad \forall b.[\overline{b}](b = n_l \wedge o \leftarrow n_r \leftarrow STOP \end{aligned}$$

It is a straightforward task to translate the above equational property descriptions into proper greatest fixed point formulas. This task is left to the reader. The main lemmas and the final correctness property (item 5) is stated in the following proposition:

Proposition 10.1

1. $\vdash_C^{()} BufCell(o,d,n_l,n_r) : Bc(o,d,n_l,n_r)$
2. $i{\neq}o, i{\neq}n, d{\neq}a, x : Sc(i,o',n',a), y : Bc(o,d,n',n) \vdash_C^{o',n'} x \mid y : Sc(i,o,n,a)$
3. $i \neq o, i \neq n \vdash_C^{()} StartCell(i,o,n) : Sc(i,o,n,a)$
4. $x : Sc(i,o,n,a) \vdash_C^{()} x : NoSpuOut'(i,o,a)$
5. $i \neq o \vdash_C^{()} GCBuf(i,o) : NoSpuOut(i,o)$

Most of these items are proved in a straightforward manner. The exception, if any, is 10.1.2. The goal-directed proof starts with the desired judgment:

$$i{\neq}o, i{\neq}n, d{\neq}a, x{:}Sc(i,o',n',a), y{:}Bc(o,d,n',n) \vdash_C^{o',n'} x \mid y : Sc(i,o,n,a). \quad (51)$$

We start by approximating and unfolding the right hand side fixed point, then introducing the conjunctions to the right. The result is the following three subgoals:

$$i{\neq}o, \cdots, x : \cdots, y : \cdots \vdash_C^{o',n'} x \mid y : [\tau]Sc(i,o,n,a) \quad (52)$$

$$i{\neq}o, \cdots, x : \cdots, y : \cdots \vdash_C^{o',n'} x \mid y : \forall b.[b]((b = i \wedge \cdots) \vee (b = n \wedge \cdots)) \quad (53)$$

$$i{\neq}o, \cdots, x : \cdots, y : \cdots \vdash_C^{o',n'} x \mid y : \forall b.[\overline{b}](b = o \wedge \cdots) \quad (54)$$

First, for (52), we observe that τ-steps of each of x or y give immediately cause for discharge. Furthermore, no communication between x and y is enabled: x may input along i or n', but y may only output along o — neither of these can be identified given the assumption and the restriction set. Also x may only output along o' but y may only input along n — again these can not be identified. Thus we can dispense with subgoal (52).

Then, for subgoal (53), there are two possibilities for input, along i and along n (and these are known to be distinct channels). First, in the case of input along i the subgoal is (after some initial reasoning steps) reduced to

$$\begin{array}{l} i{\neq}o, \cdots, x : d = a \vee Sc(i, o', n', a), y : Bc(o, d, n', n) \\ \quad \vdash_C^{o',n'} \quad x \mid y : d = a \vee Sc(i, o, n, a) \end{array} \tag{55}$$

which is resolved almost immediately. Second, in the case of input along n the subgoal is reduced to

$$\begin{array}{l} i{\neq}o'', i \neq n'', \cdots, x : Sc(i, o', n', a), y : Bc(o'', d, n', n'') \\ \quad \vdash_C^{o',n'} \quad x \mid y : Sc(i, o'', n'', a) \end{array} \tag{56}$$

which can be discharged against the top-level goal (proposition 10.1.2).

For subgoal (54), then, we need to consider output along o. We obtain the following reduced subgoal:

$$i{\neq}o, \cdots, x : Sc(i, o', n', a), y : Bc_1(o, n', n) \tag{57}$$

$$\vdash_C^{o',n'} \quad x \mid y : Sc(i, o, n, a). \tag{58}$$

Again we need to approximate and unfold the right hand-side formula, and consider each case of action type in turn. This analysis is simpler than the one we've already done except for the situation characteristic of Bc_1 of outputting along (in this case) n' to x. The result (after enacting the reductions corresponding to this scenario) in this case is a subgoal of the shape

$$i \neq o, \cdots, x : Sc(i, o, n, a), y : STOP \vdash x \mid y : Sc(i, o, n, a) \tag{59}$$

which is an instance of a generally (and easily) provable fact, that parallel composition with the $STOP$ process does not affect behaviour. We can thus regard absence of spurious output for our garbage-collecting unbounded buffer as proved.

11 Conclusion

Earlier work on modal and temporal logic and the π-calculus includes [13], [4] and [1]. The work by Milner, Parrow and Walker did not consider temporal connectives. In [4] we attempted an automated, model checking approach restricted to finite control processes. In [1] we reconsidered the model checking problem and also gave a proof system, however for non-recursive formulas only. Also in the present paper the details are very different: the temporal logic is somewhat different, the proof system is cleaner, and we use a symbolic approach which is essential for efficiency in practical applications.

There are several important lines of enquiry for future work. The first concerns the practical applicability of the proof system. The sheer number of rules involved in the proof system may seem disheartening. On the other hand most rules are actually very intuitive — given the number and nature of the basic

process and formula connectives there are just a lot of cases to be considered. Thus, for any *given* judgment, only a small and easily comprehensible collection of rules are actually applicable. Moreover by far the large majority of proof steps are entirely mechanical, and only at very specific places (choice points, cut's, applications of the (DILEMMA) rule) is intelligence required. Thus the proof system is much better geared to computer aided tools than to pen-and-paper. At SICS we are currently extending the work reported here to a core fragment of Ericsson's Erlang programming language [3] including features such as asynchronous buffered communication, data types such as natural numbers, lists, atoms, pairing and process identifiers (pid's), dynamic pid creation, process spawning, sequential composition, pattern matching.

An equally important and related line of enquiry concerns the source of incompleteness of the proof system. Intuitively the key problem is that in proving properties of a parallel composition one must guess properties of the components. But it is not always possible to find such properties as it may very well be the case that $P \mid Q$ has a property like divergence (the capability of performing an infinite sequence of internal computation steps) because P and Q have properties (context-free, context-sensitive, or beyond) that are inexpressible in our logic. It has sometimes been argued that this problem makes the entire problem of devising compositional proof systems a futile one. We do not at all subscribe to this view, however. Compositionality should not be viewed as an all-or-nothing matter, rather, compositionality is a useful tool, to be brought to bear when warranted by the specific situation. But: is expressiveness the only source of incompleteness? If this is the case we ought to be able to prove completeness, or maybe even decidability, for judgments like

$$\Gamma, x : \phi, y : \phi \vdash^s x \mid y : \gamma. \tag{60}$$

If it is not we would still like to ask whether derivability of judgments like (60) is decidable as this would have obvious implications for the utility of our approach. Results like these would be particularly important as we as yet only have examples such as the ones given to bring out our intuition that the principles we are exploring are essential improvements upon those basing themselves solely on global state exploration (including, for instance, the approach of [2]). For observe that the completeness results for finite control agents established here only suffice to show that the power of our compositional proof system is not worse than what one can achieve using much simpler global approaches such as [4].

Other issues concern the fundamentals of the proof system. The rules of discharge needs better motivation than the one we are giving here. Really one should view the discharge conditions given here as finitary approximations of conditions applying to infinite proof structures, leading to automata based characterisations. Even the shape of the local rules themselves, as well as the choice of logical connectives is open to debate. Why do we choose some connectives over others? Certainly the private input modality considered in [1] is potentially very valuable in applications. Does it make sense to devise a separate logical connective for restriction? Concerning the shape of basic judgments, how important is the use of the relativized turnstile? We use restriction sets in the style

of [1] (and [18]) in order not to violate alpha-convertibility, even in the presence of free process variables. On the other hand, expecting alpha-convertibility of open terms may be unreasonable and counterintuitive, and by abandoning this requirement judgments, and hence the entire proof system, may be simplified rather considerably.

Further, we need to obtain characterisations of expressiveness along the lines explored for the modal μ-calculus itself by a number of authors, e.g. [8]. Concerning the choice of proof rules better completeness and decidability results will give much sharper handles on the kind of rules that should be admitted. A promising approach is to embed directly into the proof system the operational semantics proof rules in the style suggested by Simpson [17] for Hennessy-Milner logic (without fixed points). First investigations in this direction for CCS and the modal μ-calculus are reported in [7]. Progress in this direction would be useful to remove some apparent arbitrariness in our choice of process calculus. Really we should expect to be able to construct similar logics and proof systems for whichever versions of π-calculus, or other calculus for that matter, one might want to come up with. Not all logics or proof systems would be equally attractive, and indeed some effort has been put into the choice of a version of the π-calculus which remains both faithful to the original π-calculus, while at the same time permitting as orthogonal treatments of the different modalities as possible. In particular we have chosen to avoid the issue of sorting. In our calculus local deadlocks may arise in case the number of arguments in sending and receiving actions does not match. What the deeper relations should be between sorting (and more generally types and static analysis techniques), and semantically based proof systems as the ones we consider here, however, is outside the scope of this paper.

Acknowledgements

Thanks are due to Roberto Amadio, Lars-åke Fredlund, and Dilian Gurov for many discussions on these and related topics. Early parts of the work were developed in collaboration with Roberto Amadio and reported in [1]. Dilian Gurov helped, among other matters, with formulating the rule of discharge. Finally, thanks are due also to Joachim Parrow for his examples of buffer specifications.

References

[1] R. Amadio and M. Dam. A modal theory of types for the π-calculus. In *Proc. FTRTFT'96,* Lecture Notes in Computer Science, 1135:347–365, 1996.

[2] Henrik Reif Andersen, Colin Stirling, and Glynn Winskel. A compositional proof system for the modal μ-calculus. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 144–153, Paris, France, 4–7 July 1994. IEEE Computer Society Press.

[3] J. Armstrong, R. Virding, C. Wikström, and M. Williams. *Concurrent Programming in Erlang (Second Edition).* Prentice-Hall International (UK) Ltd., 1996.

[4] M. Dam. Model checking mobile processes. *Information and Computation*, 129:35–51, 1996.

[5] M. Dam. Proving properties of dynamic process networks. *Information and Computation*, 140:95–114, 1998.

[6] M. Dam, L.-å. Fredlund, and D. Gurov. Toward parametric verification of open distributed systems. In *Compositionality: the Significant Difference,* H. Langmaack, A. Pnueli and W.-P. de Roever (eds.), Springer, 1536:150–185, 1998.

[7] M. Dam and D. Gurov. Compositional verification of CCS processes. In *Proc. PSI'99*, 1999.

[8] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to the monadic second order logic. In *Proc. CONCUR'94,* Lecture Notes in Computer Science, 1119:263–277, 1996.

[9] R. Milner. *Communication and Concurrency.* Prentice Hall International, 1989.

[10] R. Milner. The polyadic π-calculus: A tutorial. Technical Report ECS-LFCS-91-180, Laboratory for the Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1991.

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

[12] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40 and 41–77, 1992.

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

[14] D. Park. Finiteness is mu-Ineffable. *Theoretical Computer Science*, **3**:173–181, 1976.

[15] D. Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996.

[16] Davide Sangiorgi. From π-calculus to Higher-Order π-calculus — and back. in *Proc. TAPSOFT'93* Lecture Notes in Computer Science, 668:151–166, 1993.

[17] A. Simpson. Compositionality via cut-elimination: Hennessy-Milner logic for an arbitrary GSOS. In *Proc. LICS*, pages 420–430, 26–29 1995.

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

[19] D. Walker. Objects in the π-calculus. *Information and Computation*, 116:253–271, 1995.

Appendix A. Proofs for Section 6

Proposition 6.3 *Let N include the set of non-fresh names of $\Gamma \vdash^s E : \phi$.*

1. *If ϕ is elementary then $hyp(\eta, N) \models^s E : \phi$ if and only if $hyp(\eta, N) \vdash^s E : \phi$.*
2. *$\Gamma \vdash^s E : \phi$ if and only if $hyp(\nu s.\eta, N) \vdash^s E : \phi$ for all η such that $\nu s.\eta \models \Gamma$.*

Proof: (1) If: Use soundness. Only-if: By induction on the structure of ϕ. Let $\phi = \forall a.\psi$. Reduce $hyp(\eta, N) \vdash^s E : \forall a.\psi$ to $hyp(\eta, N) \vdash^s E : \psi\{b/a\}$ where $b \notin N$ by ($\forall$-R). Using (D-L) this is reduced, using elementary reasoning, to the set of judgments $hyp(\eta', N \cup \{b\}) \vdash^s E : \psi\{b/a\}$ where η' is required to agree with η on N. But each element in this set is provable by the induction hypothesis. Let then $\phi = \exists a.\psi$. Since $hyp(\eta, N) \models^s E : \phi$ we find a $b \in \mathcal{G}$ such that $hyp(\eta, N) \models^s E : \psi\{b/a\}$. Either b can be chosen in N in which case $hyp(\eta, N) \vdash^s E : \psi\{b/a\}$ by the ind. hyp. so that $hyp(\eta, N) \vdash^s E : \phi$ by ($\exists$-R), or else $hyp(\nu b.\eta, N \cup \{b\}) \models^s E : \psi\{b/a\}$ in which case $hyp(\nu b.\eta, N \cup \{b\}) \vdash^s E : \psi\{b/a\}$, thus also $hyp(\nu b.\eta, N \cup \{b\}) \vdash^s E : \phi$ by ($\exists$-R). But then

$$hyp(\eta, N), x : (b \neq a_1) \wedge \cdots \wedge (b \neq a_n) \vdash^s E : \phi$$

too (by definition (6.2)), and then

$$hyp(\eta, N), x : \exists b.(b \neq a_1) \wedge \cdots \wedge (b \neq a_n) \vdash^s E : \phi.$$

But then it suffices to use (Infty) and (Cut-1) to prove $hyp(\eta, N) \vdash^s E : \phi$ as desired. The remaining cases are quite easy.
(2) By induction on the complexity of Γ.
$\Gamma = ()$. If $\vdash^s E : \phi$ then $hyp(\nu s.\eta, N) \vdash^s E : \phi$ by (W-L). In the other direction use (D-L).
$\Gamma = \Gamma', x : \forall a.\psi$. Suppose first that $\Gamma \vdash^s E : \phi$ and $\nu s.\eta \models \Gamma$. Reduce the goal $hyp(\nu s.\eta, N) \vdash^s E : \phi$ to the two subgoals

$$hyp(\nu s.\eta, N) \vdash^{()} x : \forall a.\psi \tag{61}$$

$$hyp(\nu s.\eta, N), x : \forall a.\psi \vdash^s E : \phi \tag{62}$$

The second subgoal is easily dealt with using (Cut-1) and the induction hypothesis by showing that $hyp(\nu s.\eta, N) \vdash^{()} y : \gamma$ whenever $y : \gamma$ is a hypothesis in Γ'. The first subgoal is proved first using ($\forall$-R) to reduce to $hyp(\nu s.\eta, N) \vdash^{()} x : \psi\{b/a\}$ and then by (D-L) to all goals of the form $hyp(\nu s.\eta', N \cup \{b\}) \vdash^{()} x : \psi\{b/a\}$ where η' agrees with η on N. Using the induction hypothesis this can be reduced to the single goal $\Gamma', x_1 : \psi\{a_1/a\}, \ldots, x_m : \psi\{a_m/a\}, x_{m+1} : \psi\{b/a\} \vdash^{()} x : \psi\{b/a\}$ which is directly provable by (I) and (E-L). Conversely we need to show $\Gamma \vdash^s E : \phi$ from the assumption that $hyp(\nu s.\eta, N) \vdash^s E : \phi$ whenever $\nu s.\eta \models \Gamma$. First reduce $\Gamma \vdash^s E : \phi$ to

$$\Gamma', x_1 : \forall a.\psi, \ldots, x_{n+1} : \forall a.\psi \vdash^s E : \phi$$

where $N = \{a_1, \ldots, a_n\}$. Using (CUT-1) and (INFTY) this is reduced to

$$\Gamma', x_1 : \forall a.\psi, \ldots, x_n : \forall a.\psi, x_{n+1} : \forall a.\psi \wedge \exists b.b \neq a_1 \wedge \cdots \wedge b \neq a_n \vdash^s E : \phi.$$

Now, by ($\exists$-L) and ($\forall$-L) this is reduced to

$$\Gamma', x_1 : \forall a.\psi, \ldots, x_n : \forall a.\psi, x_{n+1} : \psi\{b/a\} \wedge b \neq a_1 \wedge \cdots \wedge b \neq a_n \vdash^s E : \phi$$

where b is fresh, and then using (CUT-1) and ($\exists$-R) finally to

$$\Gamma', x_1 : \forall a.\psi, \ldots, x_n : \forall a.\psi, x_{n+1} : \exists b.\psi\{b/a\} \wedge b \neq a_1 \wedge \cdots \wedge b \neq a_n \vdash^s E : \phi. \tag{63}$$

But observe that $\nu s.\eta \models \Gamma$ if and only if $\nu s.\eta \models \Gamma', x_1 : \forall a.\psi, \ldots, x_n : \forall a.\psi, x_{n+1} : \exists b.\psi\{b/a\} \wedge b \neq a_1 \wedge \cdots \wedge b \neq a_n$ whence the result follows by the induction hypothesis.
The remaining cases are left for the reader. □

Lemma 6.4 (Decomposition) *Let N include $\mathrm{fn}(E\{E_1/x_1, \ldots, E_n/x_n\})$, $\mathrm{fn}(s)$, and $\mathrm{fn}(\phi)$, suppose $E\{E_1/x_1, \ldots, E_n/x_n\}$ is closed, and suppose ϕ is non-recursive. Let $\Gamma = hyp(\nu s.\eta, N)$. If $\Gamma \models^s E\{E_1/x_1, \ldots, E_n/x_n\} : \phi$ then there are $\phi_1, \ldots, \phi_n$ of modal depth not exceeding that of ϕ such that*

1. *for all $i : 1 \leq i \leq n$, Γ, s fresh $\models^{()} E_i : \phi_i$, and*
2. *$\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \phi$*

PROOF: Assume the preconditions of the Lemma hold. The proof is by induction on the modal depth of ϕ, then structure of E, and then structure of ϕ. The first-order connectives can be dealt with in a generic manner.

$\phi = \bot$. Contradiction.

$\phi = a = b$. Let $\phi_i = \top$ for all $i : 1 \leq i \leq n$. We obtain Γ, s *fresh* $\models^{()} E_i : \phi_i$, and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \phi$ by elementary reasoning and proposition 6.3.1. The cases for ϕ an inequation $a \neq b$ or $\phi = \top$ are similar.

$\phi = \psi_1 \vee \psi_2$. For $j = 1$ or $j = 2$, $\Gamma \models^s E\{E_1/x_1, \ldots, E_n/x_n\} : \psi_j$. By the induction hypothesis we find $\phi_1, \ldots, \phi_n$ (of sufficiently small modal depth) such that Γ, s *fresh* $\models^{()} E_i : \phi_i$ for all $i : 1 \leq i \leq n$, and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \psi_j$. Then we are done by $\vee$-R.

$\phi = \forall a.\psi$. Suppose that a is not in N. We obtain $\Gamma \models^s E\{E_1/x_1, \ldots, E_n/x_n\} : \psi$. By the induction hypothesis we find appropriate $\psi_1, \ldots, \psi_n$ such that

$$\Gamma, s \text{ fresh} \vdash^{()} E_i : \psi_i \quad (\forall i : 1 \leq i \leq n)$$
$$\Gamma, x_1 : \psi_1, \ldots, x_n : \psi_n \vdash^s E : \psi$$

Let $\phi_i = \forall a.\psi_i$. Since a was chosen not in N, $\Gamma \vdash^{()} E_i : \phi_i$, and by the $\forall$-rules, $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \phi$ too. The case for $\wedge$ is similar.

$\phi = \exists a.\psi$. We find a b such that $\Gamma \models^s E\{E_1/x_1, \ldots, E_n/x_n\} : \psi\{b/a\}$, thus by the induction hypothesis we find (appropriate) $\phi_1, \ldots, \phi_n$ such that $\Gamma \vdash^{()} E_i : \phi_i$

for all i, and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \psi$. Then, by ($\exists$-R), $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \phi$ and we are done.

Secondly we can deal with some process connectives in a generic manner too:

$E =$ **if** $a = b$ **then** F_1 **else** F_2. Assume $a, b \notin s$. Assume $\Gamma \models^s E : a = b$ (other case is symmetric). Then $\Gamma \models^s F_1 : \phi$. By the 2nd level induction hypothesis we find $\phi_1, \ldots, \phi_n$ such that $\Gamma, s \; \mathit{fresh} \models^() E_i : \phi_i$ for all i, and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s F_1 : \phi$. Since $\Gamma \vdash^s F_1 : a = b$ by proposition 6.3.1, by (W-L) also $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s F_1 : a = b$. Then we use (COND) along with elementary reasoning to conclude $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \phi$ as desired. Let then (e.g.) $a \in s$ and $a \neq b$. We obtain that $\Gamma \models^s F_2 : \phi$ since in the context s, a and b can not be identified. We then use the 2nd level induction hypothesis to find $\phi_1, \ldots, \phi_n$ such that $\Gamma, s \; \mathit{fresh} \models^() E_i : \phi_i$ and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s F_2 : \phi$. By (W-L), $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n, y : a \neq b \vdash^s F_2 : \phi$, and by (NEW1), $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n, y : a = b \vdash^s F_1 : \phi$, so by (COND), $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \phi$ as desired. Remaining is the case for $a = b$ and $a \in s$. In that case we obtain $\Gamma \models^s F_1 : \phi$ so by the 2nd level induction hypothesis we find $\phi_1, \ldots, \phi_n$ such that $\Gamma, s \; \mathit{fresh} \models^() E_i : \phi_i$ and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s F_1 : \phi$. By (W-L), $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n, y : a = b \vdash^s F_1 : \phi$, and by (IRR), $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n, y : a \neq b \vdash^s F_2 : \phi$. Thus $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : \phi$ as desired.

$E = \nu a.F$. We obtain $\Gamma \models^{s,a} F : \phi$, so by the 2nd level induction hypothesis we find $\phi_1, \ldots, \phi_n$ such that $\Gamma, (s, a) \; \mathit{fresh} \models^() E_i : \phi_i$ for all i, and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^{s,a} F : \phi$, thus $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s \nu a.F : \phi$ by (NEW2). Moreover, $\Gamma, s \; \mathit{fresh} \models^() E_i : \phi_i$ too since a is fresh.

For the remaining connectives we proceed by induction first on modal depth and then on agent structure.

$\phi = <l>\psi$. We continue by induction on the structure of E.

- $E = 0$. Contradiction.
- $E = F_1 + F_2$. $\Gamma \models^s F_j : <a>\psi$ for $j = 1$ or $j = 2$. By the 2nd level induction hypothesis find $\phi_1, \ldots, \phi_n$ of desired properties. Then $\Gamma, s \; \mathit{fresh} \models^() E_i : \phi_i$ for all i, and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s E : <a>\psi$ by ($<l_\tau>$-PLUS).
- $E = \tau.F$. Contradiction.
- $E = l'.F$. In this case $\Gamma \models^s F : l = l'$ and $\Gamma \models^s F : \psi$. By the outer level induction hypothesis and ($<l>$-ACT) we are then done.
- $E = F_1 \mid F_2$. Assume $F_1 \xrightarrow{l} F_1'$ and $\Gamma \models^s F_1' \mid F_2 : \psi$ (the other case is symmetrical). By the outer level induction hypothesis we find ψ_1', ψ_2' such that $\Gamma, s \; \mathit{fresh} \models^() F_1' : \psi_1'$, $\Gamma, s \; \mathit{fresh} \models^() F_2 : \psi_2'$, and $\Gamma, y_1 : \psi_1', y_2 : \psi_2' \vdash^s y_1 \mid y_2 : \psi$. Let $\psi_1 = <l>\psi_1'$ and $\psi_2 = \psi_2'$. We obtain $\Gamma, s \; \mathit{fresh} \models^() F_1 : \psi_1$, $\Gamma, s \; \mathit{fresh} \models^() F_2 : \psi_2$, and $\Gamma, y_1 : \psi_1, y_2 : \psi_2 \vdash^s$

$y_1 \mid y_2 : <l>\psi$ by ($<l>$-PAR) and some elementary reasoning. Now use the 2nd level induction hypothesis to find $\phi_{j,1}, \ldots, \phi_{j,n}$, $j \in \{1,2\}$, such that $\Gamma, s\ fresh \models^{()} E_i : \phi_{j,i}$ for all i and j, and $\Gamma, x_1 : \phi_{j,1}, \ldots, x_n : \phi_{j,n}, s\ fresh \vdash^{()} F_j : \psi_j$. Let then $\phi_i = \phi_{1,i} \wedge \phi_{2,i}$, and the result is obtained using elementary reasoning and some cuts.

- $E = \lambda a.F$ and $E = [a]F$. Contradiction.
- $E = x$. Since $E\{a_1/x_1, \ldots, a_n/x_n\}$ is closed, $x = x_k$ for some $k : 1 \leq k \leq n$. We can then proceed by induction on the structure of E_k.

$\phi = <\tau>\psi$. Given the previous case, and observing that all rules relevant to the $<l>$ modality have correlates for $<\tau>$, the only subcase warranting attention is the case for $E = F_1 \mid F_2$. Assume first that $F_1 \xrightarrow{\overline{a}} \nu b.\langle b\rangle F_1'$ and $F_2 \xrightarrow{a} (c)F_2'$ such that $E \xrightarrow{\tau} \nu b.(F_1' \mid (F_2'\{b/c\}))$, and $\Gamma \models^{s} \nu b.(F_1'(F_2'\{b/c\})) : \psi$. Then $\Gamma \models^{s,b} F_1'(F_2'\{b/c\}) : \psi$. By the outer induction hypothesis we find ψ_1' and ψ_2' such that (since b is fresh)

$$\Gamma, (s,b)\ fresh \models^{()} F_1' : \psi_1', \quad \Gamma, (s,b)\ fresh \models^{()} F_2' : \psi_2', \tag{64}$$

$$\Gamma, y_1 : \psi_1', y_2 : \psi_2'\{b/c\} \vdash^{s,b} y_1 \mid y_2 : \psi. \tag{65}$$

Let $\psi_1 = <\overline{a}>(\nu b \leftarrow \psi_1')$ and $\psi_2 = <a>\nu b \rightarrow \psi_2'$. Then $\Gamma, s\ fresh \models^{()} F_j : \psi_j$, and the goal we need to show is

$$\Gamma, y_1 : \psi_1, y_2 : \psi_2 \vdash^{s} y_1 \mid y_2 : <\tau>\psi. \tag{66}$$

Use ($<\tau>$-PAR) to reduce this to the subgoals

$$\Gamma, y_1 : \nu b \leftarrow \psi_1', y_2 : \psi_2, s\ fresh \vdash^{()} y_1 : isCb \tag{67}$$

$$\Gamma, y_1 : \psi_1, y_2 : \psi_2', s\ fresh \vdash^{()} y_1 : isA \tag{68}$$

$$\Gamma, y_1 : \nu b \leftarrow \psi_1', y_2 : \nu b \rightarrow \psi_2' \vdash^{s} y_1 \mid y_2 : \psi. \tag{69}$$

The first two subgoals are quite trivial, and 69 is resolved by ($\nu \rightarrow$-$\nu \leftarrow$-COM) to reduce to

$$\Gamma, y_1 : \psi_1', y_2 : \psi_2' \vdash^{s,b} y_1 \mid y_2 : \psi. \tag{70}$$

Use then ($\vee$-L) and (NEW1) to obtain the subgoal

$$\Gamma, y_1 : \psi_1', y_2 : \psi_2' \vdash^{s,b} y_1 \mid y_2 : \psi \tag{71}$$

which is proved (65). The other case, where the communication between F_1 and F_2 is free, is similar and left for the reader.

$\phi = [l]\psi$. Similar to the case for $<a>$.

$\phi = [\tau]\psi$. We consider only the case for $E = F_1 \mid F_2$. We identify formulas $\psi_1, \psi_2, \psi_3, \psi_4$ for F_1 such that $\Gamma, s\ fresh \models^{()} F_1 : \psi_1 \wedge [\tau]\psi_2 \wedge \forall a.[a]\psi_3 \wedge \forall a.[\overline{a}]\psi_4$, and similarly formulas $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ for F_2 such that $\Gamma, s\ fresh \models^{()} F_2 : \gamma_1 \wedge [\tau]\gamma_2 \wedge \forall a.[a]\gamma_3 \wedge \forall a.[\overline{a}]\gamma_4$, such that the premises of ($[\tau]$-PAR) become derivable. So, assume first that $E\{E_1/x_1, \ldots, E_n/x_n\}(\nu s.\eta) \xrightarrow{\tau} F$ so that $\Gamma \models^{s} F : \psi$. There are six cases of interest:

1. $F_1(\nu s.\eta) \xrightarrow{\tau} F_1'$ such that $F = F_1' \mid F_2(\nu s.\eta)$.

2. $F_1(\nu s.\eta) \xrightarrow{\overline{a}} \langle b\rangle F_1'$ and $F_2(\nu s.\eta) \xrightarrow{a} (c)F_2'$ such that $F = F_1' \mid (F_2'\{b/c\})$.

3. $F_1(\nu s.\eta) \xrightarrow{\overline{a}} \nu b.\langle b\rangle F_1'$ and $F_2(\nu s.\eta) \xrightarrow{a} (c)F_2'$ such that $F = \nu b.(F_1' \mid (F_2'\{b/c\}))$.

Cases (4)–(6) are symmetric to the above. In case (1) we find, using the outer level induction hypothesis, $\psi(F)$ and $\gamma(F)$ such that $\Gamma, s\ \mathit{fresh} \models^{()} F_1' : \psi(F)$, $\Gamma, s\ \mathit{fresh} \models^{()} F_2 : \gamma(F)$, and $\Gamma, y_1 : \psi(F), y_2 : \gamma(F) \vdash^s y_1 \mid y_2 : \psi$. In case (2) we find $\psi(F)$ and $\gamma(F)$ such that $\Gamma, s\ \mathit{fresh} \models^{()} F_1' : \psi(F)$, $\Gamma, s\ \mathit{fresh} \models^{()} F_2'\{b/c\} : \gamma(F)$, and $\Gamma, y_1 : \psi(F), y_2 : \gamma(F) \vdash^s y_1 \mid y_2 : \psi$. In case (3) we find $\psi(F)$ and $\gamma(F)$ such that $\Gamma, s\ \mathit{fresh} \models^{()} F_1' : \psi(F)$, $\Gamma, s\ \mathit{fresh} \models^{()} F_2'\{b/c\} : \gamma(F)$, and $\Gamma, y_1 : \psi(F), y_2 : \gamma(F) \vdash^s \nu b.y_1 \mid y_2 : \psi$. In the symmetric cases $\psi(F)$ and $\gamma(F)$ are identified similarly. Define now

$$
\begin{aligned}
\psi_1 &= \bigwedge\{\psi(F_1 \mid F_2') \mid F_2(\nu s.\eta) \xrightarrow{\tau} F_2'\} \\
\psi_2 &= \bigvee\{\psi(F_1' \mid F_2) \mid F_1(\nu s.\eta) \xrightarrow{\tau} F_1'\} \\
\psi_3 &= (\mathit{isA} \supset \bigvee\{\bigwedge\{b \to \psi(F_1'\{b/c\} \mid F_2') \mid F_2(\nu s.\eta) \xrightarrow{\overline{a}} \langle b\rangle F_2'\} \mid F_1(\nu s.\eta) \xrightarrow{a} (c)F_1'\}) \\
&\quad \wedge(\mathit{isA} \supset \bigvee\{\bigwedge\{\nu b \to \psi(\nu b.F_1'\{b/c\} \mid F_2'\{b/d\}) \mid \\
&\qquad\qquad F_2(\nu s.\eta) \xrightarrow{\overline{a}} \nu d.\langle d\rangle F_2'\} \mid F_1(\nu s.\eta) \xrightarrow{a} (c)F_1'\}) \\
\psi_4 &= (\mathit{isCf} \supset \bigvee\{\bigwedge\{b \leftarrow \psi(F_1' \mid F_2'\{b/c\}) \mid F_2(\nu s.\eta) \xrightarrow{a} (c)F_2'\} \mid F_1(\nu s.\eta) \xrightarrow{\overline{a}} \langle b\rangle F_1'\}) \\
&\quad \wedge(\mathit{isCb} \supset \bigvee\{\bigwedge\{\nu b \leftarrow \psi(\nu b.(F_1'\{b/d\} \mid F_2'\{b/c\})) \mid F_2(\nu s.\eta) \xrightarrow{a} (c)F_2'\} \mid \\
&\qquad\qquad F_1(\nu s.\eta) \xrightarrow{\overline{a}} \nu d.\langle d\rangle F_1'\})
\end{aligned}
$$

where $b\ \mathit{fresh}$ abbreviates the conjunction of inequations $b \neq c$ for all c that occurs freely in F. The formulas $\gamma_1, \ldots, \gamma_3$ are defined symmetrically. We need to show that $\Gamma, s\ \mathit{fresh} \models^{()} F_1 : \psi_1 \wedge [\tau]\psi_2 \wedge \forall a.[a]\psi_3 \wedge \forall a.[\overline{a}]\psi_4$. Each conjunct is considered separately. The first two are quite easy and left aside. For the third we need to show, e.g., that

$$
\begin{aligned}
\Gamma, s\ \mathit{fresh} \models^{()} (c)F_1' : \mathit{isA} \supset \bigvee\{\bigwedge\{\nu b \to \psi(\nu b.F_1'\{b/c\} \mid F_2'\{b/d\}) \mid \\
F_2(\nu s.\eta) \xrightarrow{\overline{a}} \nu d.\langle d\rangle F_2'\} \mid F_1(\nu s.\eta) \xrightarrow{a} (c)F_1'\}.
\end{aligned}
$$

This is reduced to

$$
\Gamma, s\ \mathit{fresh} \models^{()} (c)F_1' : \forall b.b \to b\ \mathit{fresh} \supset \psi(\nu b.F_1'\{b/c\} \mid F_2'\{b/d\})
$$

for all F_2' such that $F_2(\nu s.\eta) \xrightarrow{\overline{a}} \nu d.\langle d\rangle F_2'$. So assume $F_2(\nu s.\eta) \xrightarrow{\overline{a}} \nu d.\langle d\rangle F_2'$. Assuming b fresh we then need to show

$$
\Gamma, s\ \mathit{fresh} \models^{()} F_1'\{b/c\} : \psi(\nu b.F_1'\{b/c\} \mid F_2'\{b/d\})
$$

but this follows by the induction hypothesis. The remaining conjuncts of ψ_3 and $\psi_4$are verified similarly.

We also need to show

$$\Gamma, y_1 : \psi_1 \wedge [\tau]\psi_2 \wedge \forall a.[a]\psi_3 \wedge \forall a.[\overline{a}]\psi_4, y_2 : \gamma_1 \wedge [\tau]\gamma_2 \wedge \forall a.[a]\gamma_3 \wedge \forall a.[\overline{a}]\gamma_4 \vdash^s y_1 \mid y_2 : [\tau]\psi.$$

By ($[\tau]$-PAR) we need to show the following:

$$\Gamma, y_1 : \psi_1, y_2 : \gamma_2 \vdash^s y_1 \mid y_2 : \psi \tag{72}$$
$$\Gamma, y_1 : \psi_2, y_2 : \psi_1 \vdash^s y_1 \mid y_2 : \psi \tag{73}$$
$$\Gamma, y_1 : \psi_3, y_1 : isA, y_2 : \gamma_4, y_2 : isCf \vdash^s y_1 \mid y_2 : \psi \tag{74}$$
$$\Gamma, y_1 : \psi_3, y_1 : isA, y_2 : \gamma_4, y_2 : isCb \vdash^s y_1 \mid y_2 : \psi \tag{75}$$
$$\Gamma, y_1 : \psi_4, y_1 : isCf, y_2 : \gamma_3, y_2 : isA \vdash^s y_1 \mid y_2 : \psi \tag{76}$$
$$\Gamma, y_1 : \psi_4, y_1 : isCb, y_2 : \gamma_3, y_2 : isA \vdash^s y_1 \mid y_2 : \psi \tag{77}$$

along with a few other subgoals which are easily proven from the assumptions. To show (72) use first ($\vee$-L) and then ($\wedge$-L) to reduce to a goal of the form $\Gamma, y_1 : \psi(F_1 \mid F_2'), y_2 : \gamma(F_1 \mid F_2') \vdash^s y_1 \mid y_2 : \psi$ which is provable by the induction hypothesis. (73) is proved similarly. To show (75) reduce the goal, using elementary reasoning, to a subgoal of the form

$$\begin{gathered}\Gamma, y_1 : \bigvee\{\bigwedge\{\nu b \to \psi(F_1'\{b/c\} \mid F_2') \mid F_2(\nu s.\eta) \xrightarrow{\overline{a}} \langle b\rangle F_2'\} \mid \\ F_1(\nu s.\eta) \xrightarrow{a} (c)F_1'\}, \\ y_2 : \bigvee\{\bigwedge\{\nu b \leftarrow \gamma(F_1' \mid F_2'\{b/c\}) \mid F_1(\nu s.\eta) \xrightarrow{a} (c)F_1'\} \mid F_2(\nu s.\eta) \xrightarrow{\overline{a}} \nu d.\langle b\rangle F_2'\} \\ \vdash^s y_1 \mid y_2 : \psi.\end{gathered}$$

To show this let $F_1(\nu s.\eta) \xrightarrow{a} (c)F_1'$ and $F_2(\nu s.\eta) \xrightarrow{a} \nu d.\langle d\rangle F_2'$, and, using ($\vee$-L), we have to show

$$\begin{gathered}\Gamma, y_1 : \bigwedge\{\forall b.b \to b\ fresh \supset \psi(F_1'\{b/c\} \mid F_2') \mid F_2(\nu s.\eta) \xrightarrow{\overline{a}} \langle b\rangle F_2'\}, \\ y_2 : \bigwedge\{\nu b \leftarrow \gamma(F_1' \mid F_2'\{b/c\}) \mid F_1(\nu s.\eta) \xrightarrow{a} (c)F_1'\} \\ \vdash^s y_1 \mid y_2 : \psi.\end{gathered}$$

This is then reduced, using ($\wedge$-L) to

$$\Gamma, y_1 : \nu b \to \psi(F_1'\{b/c\} \mid F_2'), y_2 : \nu b \leftarrow \gamma(F_1' \mid F_2'\{b/c\}) \vdash^s y_1 \mid y_2 : \psi$$

which is in turn reduced, using ($\nu \to$-$\nu \leftarrow$-COM), to

$$\Gamma, y_1 : \psi(F_1'\{b/c\} \mid F_2'), y_2 : \gamma(F_1' \mid F_2'\{b/c\}) \vdash^{s,b} y_1 \mid y_2 : \psi$$

which is provable by the induction hypothesis. Now it just remains, using the 2nd level induction hypothesis, and observing that modal depth has not increased, to identify the desired $\phi_1, \ldots, \phi_n$ and to put the desired proof together using (CUT-1). This is routine.

$\phi = a \to \psi$. Again we proceed by induction on agent structure.

- First we deal in one blow with any E which is not an abstraction since these cases are contradictory. The remaining cases are:

- $E = F_1 \mid F_2$. The only relevant case is when F_1 is an abstraction and F_2 is a process (or symmetrically). We can write F_1 as $(b)F_1'$. Then $\Gamma \models^s F_1'\{a/b\} \mid F_2 : \psi$. Since $F_1'\{a/b\}$ is of smaller size than E the 2nd level induction hypothesis applies to produce ψ_1, ψ_2 such that $\Gamma, s\ \mathit{fresh} \models^{()} F_1'\{a/b\} : \psi_1$ and $\Gamma, s\ \mathit{fresh} \models^{()} F_2 : \psi_2 \wedge \mathit{isP}$, and $\Gamma, y_1 : \psi_1, y_2 : \psi_2 \wedge \mathit{isP} \vdash^s y_1 \mid y_2 : \psi$. Now $\Gamma, s\ \mathit{fresh} \models^{()} F_1 : a \to \psi_1$ and $\Gamma, y_1 : a \to \psi_1, y_2 : \psi_2 \wedge \mathit{isP} \vdash^s y_1 \mid y_2 : a \to \psi$ as desired, by ($\to$-PAR) and a little elementary reasoning.

- $E = (b)F$. Then $\Gamma \models^s F\{E_1/x_1, \ldots, E_n/x_n, a/b\} : \psi$ so by the 2nd level induction hypothesis we find $\phi_1, \ldots, \phi_n$ such that $\Gamma, s\ \mathit{fresh} \models^{()} E_i : \phi_i$ for all i, and $\Gamma, x_1 : \phi_1, \ldots, x_n : \phi_n \vdash^s F\{a/b\} : \psi$. Then we are done by ($\to$-IN).

- $E = x$. In this case $x = x_k$ and we can proceed by induction in the structure of E_k.

$\phi = a \leftarrow \psi$, $\phi = \nu a \leftarrow \psi$, or $\phi = \nu a \to \psi$. Similar to previous case. □

Appendix B. Proof of Theorem 8.7

Theorem 8.7 *If* $\Gamma \vdash_{MC} E : \phi$ *then* $\Gamma \vdash_C E : \phi$.

PROOF: Observe first that it is safe to assume that, up to names that are not fresh, Γ determines a unique η such that $\eta \models \Gamma$. If this is not the case then the dilemma rules apply on both sides to reduce the problem to a number of problems that do possess this property. Now, since η is uniquely determined we might as well assume that η is the identity on names. Let then η be the substitution $\{E_1/x_1, \ldots, E_n/x_n\}$. Consider a model checker node v labelled

$$\Gamma \vdash \nu s.E\eta : \phi, \tag{78}$$

where the variables $x_1, \ldots, x_n$ occurs linearly in E. The aim is to produce a proof tree of $\Gamma, \Delta \vdash^s E : \phi$ where $\Delta = x_1 : C(E_1, \eta), \ldots, x_n : C(E_n, \eta)$, and show that, for the tree constructed from the root model checker node, all non-axiom leaves can be discharged. The proof is by induction on the number of occurrences of $|$ in $E\eta$, and then following the structure of the model checker proof. Most subcases of the base and inductive steps for the outer induction are common so we proceed directly to a case analysis on the structure of the model checker proof.

Suppose first that v is an axiom leaf, i.e. an instance of (I), (REFL), (IRR), (NEW1), (INFTY), or (NEC) (where, in the last case, there is no E' such that $E\eta \xrightarrow{l_\tau} E'$). In this case we obtain $\Gamma \vdash^s_C E\eta : \phi$ by theorem 6.5.

Leaves that are discharged are not considered until later.

So, assume that v is an internal node. We consider each possible rule in turn. There is nothing to do until we get to the rules that are unique to the model checker proof system.

(POSS) Assume that $\phi = <l_\tau>\phi'$ and that the model checker rule application infers $\Gamma \vdash \nu s.E\eta : <l_\tau>\phi'$ by showing that there is an E' such that $\nu s.E\eta \xrightarrow{l_\tau} E'$, and $\Gamma \vdash E' : \phi'$. Observe that E' will have the form $\nu s.E''$. We proceed by induction on size of inference that $\nu s.E\eta \xrightarrow{l_\tau} E'$ and then by cases on E.

1. E is a variable x_i. In this case $\nu s.E_i \xrightarrow{l_\tau} E'$ and we obtain (by lemma 8.6) that $\Gamma, \Delta \vdash^s_C E : \phi$.

2. (SUM). Let $E = F_1 + F_2$. If $\nu s.E\eta \xrightarrow{l_\tau} E'$ then $\nu s.F_j\eta \xrightarrow{l_\tau} E'$ for $j = 1$ or $j = 2$, and then by the innermost induction hypothesis, $\Gamma, \Delta \vdash^s_C F_j : \phi$, so by ($<l_\tau>$-PLUS), $\Gamma, \Delta \vdash^s_C E : \phi$.

3. (PRE). Let $E = l_\tau.F$. In this case $\Gamma \vdash \nu s.F\eta : \phi'$, so by the (2nd or 3rd) level induction hypothesis, $\Gamma, \Delta \vdash^s_C F : \phi'$, whence by ($(\tau)$-TAU) or ($<l>$-ACT), $\Gamma, \Delta \vdash^s_C E : \phi$.

4. (PAR). Let $E = F_1 \mid F_2$. Assume for simplicity that $s = ()$ and that the inference of $E\eta \xrightarrow{l_\tau} E'$ has the shape

$$\frac{F_1\eta \xrightarrow{\tau} F_1'}{(F_1 \mid F_2)\eta \xrightarrow{\tau} F_1' \mid (F_2\eta)}$$

(such that $E' = F_1' \mid (F_2\eta)$). To build up the proof tree from the node $\Gamma, \Delta \vdash^s E : \phi$ first apply (CUT-1) to reduce to the following two subgoals:

$$\Gamma, \Delta \vdash^{()} F_1 : Char(F_1\eta, \eta) \tag{79}$$

$$\Gamma, \Delta, x : Char(F_1\eta, \eta) \vdash^{()} x \mid F_2 : \phi \tag{80}$$

Now, by lemma 8.5 we know that $\Gamma \vdash_{MC} F_1\eta : Char(F_1\eta, \eta)$ and so, by the outermost induction hypothesis,

$$\Gamma, \Delta \vdash_C F_1 : Char(F_1\eta, \eta) \tag{81}$$

too. Thus only (80) is left. By unfolding the lhs fixed point, and by introducing $\wedge$ to the left (80) is reduced to

$$\Gamma, \Delta, x : <\tau> Char(F_1', \eta) \vdash^{()} x \mid F_2 : \phi \tag{82}$$

which in turn is reduced to

$$\Gamma, \Delta, x : Char(F_1', \eta) \vdash^{()} x \mid F_2 : \phi' \tag{83}$$

along with the subgoal $\Gamma, \Delta \vdash^{()} F_2 : isP$ which is proved by theorem 6.5. The goal (83) stands in the desired relation to the model checker node $\Gamma \vdash E' : \phi'$.

The remaining cases for (Par) are easily proved in a similar fashion.

5. (COM) — free communication. Let $E = F_1 \mid F_2$. Again we assume that $s = ()$ and that the inference of $E\eta \xrightarrow{l_\tau} E'$ has the shape

$$\frac{F_1\eta \xrightarrow{\overline{a}} \langle b\rangle F_1' \quad F_2\eta \xrightarrow{a} (d)F_2'}{E\eta \xrightarrow{\tau} F_1' \mid (F_2'\{b/d\})}$$

such that $l_\tau = \tau$. First use Lemma 8.5 to obtain

$$\Gamma \vdash_{MC} F_1\eta : Char(F_1\eta, \eta) \tag{84}$$

$$\Gamma \vdash_{MC} F_2\eta : Char(F_2\eta, \eta) \tag{85}$$

Let

$$\begin{aligned} \psi_1 &= Char(\langle b\rangle F_1', \eta) \\ \psi_2 &= Char((d)F_2', \eta). \end{aligned}$$

The goal is to prove

$$\Gamma, \Delta \vdash^{()} F_1 \mid F_2 : <\tau>\phi'. \tag{86}$$

Use two cuts to reduce to

$$\Gamma, \Delta \vdash^{()} F_1 : <\overline{a}>\psi_1 \tag{87}$$

$$\Gamma, \Delta \vdash^{()} F_2 : <a>\psi_2 \tag{88}$$

$$\Gamma, \Delta, y_1 : <\overline{a}>\psi_1, y_2 : <a>\psi_2 \vdash^{()} y_1 \mid y_2 : <\tau>\phi'. \tag{89}$$

Subgoals (87) and (88) are obtained, as before, by the outermost induction hypothesis. By ($<\tau>$-PAR), (89) is reduced to the subgoal

$$\Gamma, \Delta, y_1 : \psi_1, y_2 : \psi_2 \vdash^{()} y_1 \mid y_2 : \phi'. \tag{90}$$

We are now almost done except that the communication needs to be completed. Observe that

$$\begin{aligned} \psi_1 &= b \leftarrow \mathit{Char}(F_1', \eta) && (91)\\ \psi_2 &= \forall d.d \rightarrow \bigvee\{d = e \wedge \mathit{Char}(E\{e/d\}, \eta) \mid e \in \mathit{fn}((d)F_2')\} \vee && (92)\\ &\quad (\bigwedge\{d \neq e \mid e \in \mathit{fn}((d)F_2')\} \wedge \mathit{Char}(F_2', \nu d.\eta)) && (93) \end{aligned}$$

We now reduce (90), using ($\forall$-L), to

$$\Gamma, \Delta, y_1 : \psi_1, y_2 : b \rightarrow \bigvee\{b = e \cdots\} \vee (\cdots) \vdash^{()} y_1 \mid y_2 : \phi' \tag{94}$$

and then, by ($\rightarrow$-$\leftarrow$-COM) and boolean reasoning, to

$$\Gamma, \Delta, y_1 : \mathit{Char}(F_1', \eta), y_2 : \mathit{Char}(F_2'\{b/d\}, (\nu b.)\eta) \vdash^{()} y_1 \mid y_2 : \phi' \tag{95}$$

where the νb is present in the case b is provably distinct from all free names in $(d)F_2'$. If the νb is absent we are done immediately, and if not we only have to observe that $\mathit{Char}(F_2', \nu b.\eta) = \mathit{Char}(F_2', \eta)$, and we are done.

6. (COM) — bound communication. Let $E = F_1 \mid F_2$. Assume again that $s = ()$ and that the inference of $E\eta \xrightarrow{\tau} E'$ has the shape

$$\frac{F_1\eta \xrightarrow{\overline{a}} \nu b.\langle b\rangle F_1' \quad F_2\eta \xrightarrow{a} (d)F_2'}{E\eta \xrightarrow{\tau} \nu b.(E_1' \mid (E_2'\{b/d\}))}$$

The proof follows the previous subcase quite closely. Let

$$\begin{aligned} \psi_1 &= \mathit{Char}(\nu b.\langle b\rangle F_1', \eta)\\ \psi_2 &= \mathit{Char}((d)F_2', \eta). \end{aligned}$$

The goal $\Gamma, \Delta \vdash^{()} F_1 \mid F_2 : <\tau>\phi'$ is reduced, using two cuts and the outermost induction hypothesis to a judgment of the form (89). As the previous subcase this is further reduced to a subgoal of the form (90). Observe now that

$$\psi_1 = \nu b \leftarrow \mathit{Char}(F_1', \eta) \tag{96}$$

while ψ_2 is unchanged from (92). Now, instead of using ($\rightarrow$-$\leftarrow$-COM) we use ($\rightarrow$-ν $\leftarrow$-COM) (observe that we can use alpha-conversion to identify the bound b in ψ_1 with the bound d in ψ_2) to reduce the current goal to the following:

$$\begin{aligned} &\Gamma, \Delta, y_1 : \mathit{Char}(F_1'\{d/b\}, \nu d.\eta), y_2 : \bigvee\{d = e \wedge \cdots\} \vee (\bigwedge\{d \neq e \mid \cdots\} \wedge \cdots)\\ &\quad \vdash^{d} \quad y_1 \mid y_2 : \phi' \end{aligned} \tag{97}$$

and then further, using boolean reasoning, to

$$\Gamma, \Delta, y_1 : \mathit{Char}(F_1'\{d/b\}, \nu d.\eta), y_2 : \mathit{Char}(F_2', \nu d.\eta) \vdash^{d} y_1 \mid y_2 : \phi' \tag{98}$$

which is the required result.

7. (Res), (Open). Let $E = \nu a.F$ and assume that the inference of $E\eta \stackrel{l_\tau}{\to} E'$ has the shape

$$\frac{F\eta \stackrel{l_\tau}{\to} F'}{E\eta \stackrel{l_\tau}{\to} E'}$$

where $l_\tau \neq a$ and $l_\tau \neq \overline{a}$, and where $E' = \nu a.F'$. The reduction is simple, of $\Gamma, \Delta \vdash^{()} \nu a.F : \phi$ to $\Gamma, \Delta \vdash^{a} F : \phi$ using (New2).

8. (Id). Let $E = D(\vec{b})$ and assume that the inference of $E\eta \stackrel{l_\tau}{\to} E'$ has the shape

$$\frac{P\eta\{\vec{b}/\vec{a}\} \stackrel{l_\tau}{\to} E'}{E\eta \stackrel{l_\tau}{\to} E'}$$

because $D(\vec{a}) \stackrel{\Delta}{=} P$. In this case the reduction is straightforward from (Fix).

This completes the case for ϕ a diamond formula. The case for ϕ a box formula is quite similar, as are the cases for input/output. These cases are therefore omitted. We thus need to pay attention to the proper discharge of hypotheses. These, however, are dealt with quite simply by observing that, as the structure of the model checker proof, and in particular the pattern of unfoldings of greatest fixed point formulas, is reflected in the structure of the compositional proof system proof, and in particular leaves discharged in the former proof will correspond to leaves that can be discharged in the latter. □