Math. Struct. in Comp. Science (1999), *vol.* 9, *pp.* 63–101. Printed in the United Kingdom

A filter model for mobile processes[†]

FERRUCCIO DAMIANI, MARIANGIOLA DEZANI-CIANCAGLINI and PAOLA GIANNINI

Dipartimento di Informatica, Università di Torino
Corso Svizzera 185 – 10149 Torino (Italy)
E-mail: {damiani,dezani,giannini}@di.unito.it

Received 30 June 1997; revised 19 November 1997

This paper presents a filter model for π-calculus and shows its full abstraction with respect to a 'may' operational semantics. The model is introduced in the form of a type assignment system. Types are related by a preorder that mimics the operational behaviour of terms. A subject expansion theorem holds. Terms are interpreted as filters of types: this interpretation is compositional. The proof of full abstraction relies on a notion of realizability of types and on the construction of terms, which test when an arbitrary term has a fixed type.

1. Introduction

Process algebras are among the most successful formalisms for reasoning about concurrent processes. The starting point for the study of process algebras is the CCS formalism, introduced in Milner (1980) (see also Milner (1989) and Hennessy (1988)). Inspired by the λ-calculus (Barendregt 1984), CCS is based on a small set of basic actions and primitive constructors that are meant to characterize the behaviour of communicating processes.

Even though CCS allows for dynamic creation of processes, the topology of the possible interactions between processes is fixed by their syntax. So *mobile* systems, that is, systems with a dynamically changing pattern of communications, cannot be expressed.

Various extensions have been proposed to overcome this problem. On the one hand there is the π-calculus (Milner *et al.* 1992), see also Milner (1991), Milner (1992) and Sangiorgi (1992). In the π-calculus, channels are passed as values in communications. The restriction operator models static binding in the sense that:

1 the property of being a bound or a free channel is invariant under reduction of terms, and
2 each bound channel refers to the same binder during reduction.

On the other hand there are various versions of higher-order process calculi. In these languages, processes are explicitly passed as values in communications. Process-passing languages have been studied with respect to both the static and the dynamic binding

† This work was partly supported by NATO Grant HTECH.LG960875.

of channels. For example, Plain CHOCS (Thomsen 1990), and HOπ (Sangiorgi 1992) have static binding. In contrast, CHOCS (Thomsen 1990), and the languages studied in Hennessy (1994a) and Hennessy (1994b) have dynamic binding.

The semantics of CCS and its extensions was originally given in an operational way by bisimulation techniques.

Milner *et al.* (1993) presents a characterization of early and late bisimulation for the π-calculus via modal logics. The authors introduce a set of modal formulas and a satisfaction relation for processes based on the labelled transition system. Such satisfaction relation produces an axiomatic semantics. Amadio and Dam (1996) extends this axiomatic approach using a richer logic. The logical language considered (a version of the μ-calculus) is shown to be sufficiently expressive to characterize the finite behaviour of processes. Moreover, the authors present an algorithm that checks that processes satisfy a specification expressed in the logical language. Their proof system is quite similar to ours. The main advantage of our approach is that it gives at the same time a complete proof system and a fully abstract model.

In De Nicola and Hennessy (1984), and more extensively in Hennessy (1988), operational semantics for CCS based on 'may' and 'must' preorders are introduced. The 'may' and 'must' operational semantics test, respectively, the possibility and the necessity of performing communications. Moreover, Hennessy (1988) presents a denotational semantics that is fully abstract with respect to these operational preorders.

Denotational semantics can be given through a 'logical' description of domains. Such an approach goes back to Scott (1982), and has been advocated in Abramsky (1991a) as a general paradigm unifying, among other things, type assignment, logic of programs and logical characterizations of behaviour of concurrent processes such as Hennessy–Milner logic. The semantics can be introduced through a type assignment system, with types ordered by an inclusion relation. The interpretation of a term is the filter of types that can be assigned to it. This produces a denotational model in the sense that the denotation of a term is given by means of the denotations of its components. This has been used in Barendregt *et al.* (1983) and Abramsky and Ong (1991) for λ-calculus. Along the same lines are the studies concerning extensions of λ-calculus by means of operators with concurrent features such as Ong (1993), Boudol (1994), Dezani *et al.* (1996) and Dezani *et al.* (1997).

Abramsky (1991b) gives a denotational model of SCCS based on a domain of synchronization trees. This domain is described through a domain logic related to it by Stone duality. Hennessy (1994a) presents the first denotational model of higher-order concurrent processes based on a compromise between type systems and modal logic. Here type-formulas are built using arrows, (implicit) conjunction and prefixing connectors, expressing the ability to communicate (they are a kind of indexed modalities as in Hennessy–Milner logic). The resulting filter model turns out to be fully abstract with respect to an operational semantics based on a notion of testing and 'may' convergency. This semantics is equivalent to trace semantics. The restriction operator on channels models dynamic binding. Indeed, a process sent from one process to another can escape a restriction or can be captured by a restriction. Hennessy (1994b) builds a filter model for higher-order processes that is adequate but not complete with respect to the 'must' testing. The scoping of variables is again dynamic.

One of the important issues for process algebras is the study of the possible behaviourial equivalences. π-calculus processes are usually compared either by bisimulation (Milner *et al.* 1993) or by testing (Boreale and De Nicola 1995). In the case of bisimulation, one needs to distinguish between early and late bisimulation, reflecting the fact that an input transition may or may not correspond to one atomic event, see Milner *et al.* (1993). A well-known problem shared by early and late bisimulation, and testing for π-calculus is that the equivalence between processes obtained is not a congruence. In Boreale and Sangiorgi (1996) some general results about the limitations that must be imposed on the language to show that some bisimilarities are congruences are studied. In particular, they prove that in order to turn ground bisimulation into a congruence matching must be removed, and either the output prefix must be limited or the bisimilarity has also to take into account the causality of actions. The main problem arises from matching. For example, we have that $[x = y]\bar{a}b.\mathbf{0}$ is equivalent to $\mathbf{0}$, since both processes offer no communication, however $a(x).[x = y]\bar{a}b.\mathbf{0}$ differs from $a(x).\mathbf{0}$. In fact, $a(x).[x = y]\bar{a}b.\mathbf{0}$ can communicate with $\bar{a}y.\mathbf{0}$ yielding the process $\bar{a}b.\mathbf{0}$, while only $\mathbf{0}$ is the result of the communication between $a(x).\mathbf{0}$ and $\bar{a}y.\mathbf{0}$. To overcome this problem, Sangiorgi (1996) proposes *open bisimulation*, which compares processes under all possible substitutions. With the substitution assigning x to y (or *vice versa*) we can discriminate between $\mathbf{0}$ and $[x = y]\bar{a}b.\mathbf{0}$, which becomes $\bar{a}b.\mathbf{0}$. More generally, open bisimulation induces an equivalence between processes that is a congruence. Along similar lines, we define our observational equivalence. That is, we test processes allowing first substitutions, which may satisfy the matchings involved. In this way we obtain a congruence. In our model the interpretation of a process P is bigger than the interpretation of its restriction with respect to any channel name, that is, νxP for all x. This would no longer be true for a language including the mismatch operator (as shown in Boreale and De Nicola (1995)).

The calculus studied in this paper is the version of π-calculus with replication of processes (instead of constants) and matching, and without the 'sum' (external choice) operator. We define a 'may' operational semantics and a logically characterized denotational semantics. In particular, the denotation of a term is the filter of types (properties) assignable to the term. This interpretation is compositional in the sense that the interpretation of a term (the filter of types that can be assigned to it) is uniquely determined by the interpretation of its subterms. It is not sensible to add the external choice to our language, since the π-calculus can simulate the internal choice (as defined in De Nicola and Hennessy (1987)) and since the internal and external choices have an identical behaviour in a 'may' perspective (Milner 1980). The main problem we solved is the interpretation of the restriction on channels, since restriction obeys a static binding discipline. We quote from Hartonas and Hennessy (1997):

'Thus the major challenges are to extend the mathematical framework so that a proper scoping of channel names can be properly modelled ...'

Unfortunately, the present paper did not meet this challenge, since it gives a program logic modelling static binding, but without providing the underlying domain construction.

The contribution of this paper is methodological. We show that the technique of providing a logical model for a language based on filters of assignable properties can be applied also to languages, like the π-calculus, that include channel communication and static scoping of names. For this reason, our priority has been to keep both the process language and the operational semantics as simple as possible. So we considered the 'may' testing and a restricted version of the π-calculus that does not include many interesting process constructors, notably the mismatch. As a matter of fact, following the approach of Hennessy (1994b), we can build a filter model that is adequate with respect to the 'must' testing. To be coherent with the present approach, a logic describing must behaviours needs two type constructors, to model, respectively, the internal and the external choices. As shown in Dezani *et al.* (1996) and Dezani *et al.* (1997), the union of types corresponds in a natural way to the internal choice in a must perspective. The type constructor mimicking the external choice would be rather '*ad hoc*'. Instead, there are problems (already discussed in Hennessy (1994b)) in obtaining a complete model in this case. Essentially, test terms (see Definition 5.15) would require a quite unnatural extension of the π-calculus. We conjecture that we can also model mismatch by adding environments (mirrored by basis in the type assignment system) that state equalities and disequalities of names. Note that the above extensions would considerably increase the (already non-elementary) treatment given here.

Types express operational behaviours of processes, so our type inference system is a proof system to argue about properties of programs. Since the model defined is fully abstract, this proof system is both correct and complete. Terms not containing replication have a principal type that is dependent on their syntax, so we can define algorithms checking the observational behaviours of terms.

The definition of the language and its operational semantics are given in Section 2. The operational semantics is presented in two stages. First we define the reduction relation. Then we define a 'may' convergency predicate that expresses the capability of a process offering a communication along a given channel. We consider an observational preorder based on testing enriched by substitutions.

In Section 3 we introduce the type syntax and the preorder on types. The channel restriction corresponds in a natural way to a binding operator on channels and the matching operator of processes to a matching operator of types. The matching operator plays a central role in the definition of the equivalence between types (properties) and in the full abstraction result. Types form a preordered structure whose filters are meant to be the elements of a domain, ordered by subset inclusion.

Section 4 gives the type assignment system and proves its properties.

In Section 5 we present an interpretation of terms as the filters of types that can be assigned to them in the type assignment system. This is the logical semantics of our calculus. We show, by means of a realizability interpretation and test terms, that we can characterize the convergency properties of terms by means of the types that can be deduced for them. This implies the full abstraction of our semantics, that is, that the inclusion between interpretations of terms coincides with the 'may' observational preorder.

Section 6 contains some further comparisons with related papers.

2. The language

In this section we introduce the π-calculus without the 'sum' operator. This choice was justified in the introduction and will be discussed further in Remark 2.10.

The most primitive entity of π-calculus is a *name* or *channel*. Names, ranged over by x, y, z, …, and a, b, … have no structure. The class Π of *π-calculus processes*, ranged over by P, Q, R, M, N, …, is built from names and the null process $\mathbf{0}$ using the operators of prefixing, parallel composition, restriction, matching and replication, as follows.

Definition 2.1. Let $\mathscr{N}$ be an infinite set of *names*. Processes are defined by the following grammar

$$M ::= \mathbf{0} \;\Big|\; \alpha.M \;\Big|\; M|M \;\Big|\; \nu x\, M \;\Big|\; [x = y]M \;\Big|\; !M$$

where α is called *prefix* and can be either an *input* or an *output* :

$$\alpha ::= x(y) \;\Big|\; \bar{x}y.$$

The symbol $\mathbf{0}$ denotes the inactive process. An input-prefixed process $x(y).M$ waits for a name on the channel x and then behaves like the process M in which y is substituted by the name received. An output-prefixed process $\bar{x}y.M$ sends the names y on the channel x and then behaves like the process M. The matching $[x = y]M$ is used to test the equality of x and y. Such a process behaves like M when x is equal to y and it is inactive otherwise. The restriction construct $\nu x\, M$ declares x as a new name local to M. The parallel composition $P|Q$ means that P and Q are concurrently active, so they can act independently and also communicate. The replication $!M$ means $M|M|\cdots$, an arbitrary number of times. So there can be an arbitrary number of M executing concurrently.

Free names of a term M, $fn(M)$, are defined in the usual way, taking into account that, in both the input-prefixed process $y(x).M$ and in the restriction $\nu x\, M$, the name x is local to M, and bound in the whole term. In the following we shall be working modulo α-conversion. We can assume that bound names are different from free names and from each other (this can always be achieved by α-conversion).

We say that a process M is *mono* if and only if $fn(M)$ contains at most one name.

The substitution of x for y in P is denoted by $P\{x/y\}$, that is, $P\{x/y\}$ denotes the term obtained by replacing x for all the free occurrences of y in P. A suitable renaming of bound names is intended to avoid capturing the free name x.

We use $\overrightarrow{[a = b]}$ to denote a sequence of matchings $[a_1 = b_1] \cdots [a_m = b_m]$ $(m \geqslant 0)$, $\overrightarrow{\nu c}$ to denote a sequence of restrictions $\nu c_1 \cdots \nu c_n$ $(n \geqslant 0)$. Moreover, $x \notin \overrightarrow{[a = b]}$ is short for $x \notin \{a_1, \ldots, a_m, b_1, \ldots, b_m\}$, and $x \notin \vec{c}$ is short for $x \notin \{c_1, \ldots, c_n\}$.

A *context* is a term with a hole $[\,\cdot\,]$. Contexts will be denoted by $C[\,\cdot\,]$, and $C[M]$ is the term obtained by filling the hole with M.

We adopt the following precedence among syntactic forms, in decreasing order: substitution precedes restriction, prefixing, matching, and replicator, which all precede parallel composition.

Now, following Milner (1992) and Sangiorgi (1992), we give the operational semantics of our calculus. We first define the structural congruence between terms, and then the reduction rules for terms.

Definition 2.2. The *structural congruence* between terms, $\equiv$, is the smallest congruence that satisfies:

1 $P \equiv Q$ if P is α-convertible to Q
2 Abelian monoid laws for $|$:
$P|\mathbf{0} \equiv P$
$P|Q \equiv Q|P$
$P|(Q|R) \equiv (P|Q)|R$
3 $\nu x\, \mathbf{0} \equiv \mathbf{0}$
$\nu x\, \nu y\, P \equiv \nu y\, \nu x\, P$
$\nu x\, P|Q \equiv \nu x\, (P|Q)$ if $x \notin Q$
4 $[x = x]P \equiv P$
$!P \equiv P|!P$.

We now give the reduction rules of the calculus.

Definition 2.3. The reduction relation $\longrightarrow$ is the smallest relation that satisfies:

$$\text{COM}: \quad x(y).P|\bar{x}z.Q \longrightarrow P\{z/y\}|Q$$

$$\text{PAR}: \quad \frac{P \longrightarrow P'}{P|Q \longrightarrow P'|Q}$$

$$\text{RES}: \quad \frac{P \longrightarrow P'}{\nu x\, P \longrightarrow \nu x\, P'}$$

$$\text{STRUCT}: \quad \frac{Q \equiv P \quad P \longrightarrow P' \quad P' \equiv Q'}{Q \longrightarrow Q'}$$

We define $\longrightarrow\!\!\!\!\rightarrow$ to be $\overset{+}{\longrightarrow} \cup \equiv$, where $\overset{+}{\longrightarrow}$ is the transitive closure of $\longrightarrow$.
Since $\equiv$ is a congruence, if $M \longrightarrow\!\!\!\!\rightarrow N$, there are M_i, $1 \leqslant i \leqslant n$ such that

$$M \equiv M_1 \longrightarrow M_2 \equiv M_3 \longrightarrow \cdots \longrightarrow M_n \equiv N.$$

The restriction operator ν models static binding on names. In fact, if two processes are willing to communicate, and one of them has a channel bound by ν, first this binding is extended to both processes using the third congruence of Definition 2.2(3) and then the communication may be established (see also the discussion after Definition 2.4).

The behaviour of processes that we analyze is their ability to offer input or output communications on a given channel. This is the 'may' convergency (Hennessy 1988; Hennessy 1994a). This convergency predicate says whether there is a possible evolution of the process that leads to an offer of a communication on a given channel.

Definition 2.4. The *may* convergency predicates are defined by:

— $M \Downarrow_x^{\mathrm{may}}$ if $M \longrightarrow\!\!\!\!\rightarrow \overrightarrow{\nu c}\, (x(y).P|Q)$ for some $\vec{c}$, y, P, Q, where $x \notin \vec{c}$
— $M \Downarrow_{\bar{x}}^{\mathrm{may}}$ if $M \longrightarrow\!\!\!\!\rightarrow \overrightarrow{\nu c}\, (\bar{x}y.P|Q)$ for some $\vec{c}$, y, P, Q, where $x \notin \vec{c}$.

The previous definition is motivated by the observation that the behaviour of a parallel composition is the sum of the behaviours of its components plus the possible interaction between them. So, in considering the may convergency predicates we have to consider reductions to parallel processes one of whose components offers the required communication. Moreover, a process $\nu c\, x(y).N$, where $c \neq x$, may offer an (input) communication on x. In fact if we consider a process that does an output on x, say $\bar{x}z.M$, then

$$(\nu c\, x(y).N)|\bar{x}z.M$$

with the third congruence of Definition 2.2(3) gives

$$\nu c\, (x(y).N|\bar{x}z.M),$$

which causes the process to offer the input communication, and thus the presence of a possible restriction in the reduced term.

We compare processes through *open testing*, that is, a preorder based on the idea of considering processes under all possible substitutions on names. We call it open testing preorder since it is similar to open bisimulation (Sangiorgi 1996). In the present setting, a *tester*[†] is any process E that may use a new distinguished name w and such that every occurrence of w in E is in a subterm of the shape $\nu v\, \bar{w}v.\mathbf{0}$. To test a process M means to put a substitution instance of M in parallel with a tester. We say that the test is successful if it offers the output communication $\bar{w}$. The condition above on the occurrences of w in E expresses the fact that the name w can only be used to report success.

Definition 2.5. (Open testing preorder) Let M, N be processes. $M \sqsubseteq_{OT} N$ if for every tester E, for every substitution on names s,

$$s(M)|E \Downarrow_{\bar{w}}^{\mathrm{may}} \textit{ implies } s(N)|E \Downarrow_{\bar{w}}^{\mathrm{may}} .$$

Two other standard operational preorders (Hennessy 1994a) are defined by means of contexts and testers. The first is the observational preorder, which is based on the notion of distinguishing two terms if there is a context in which they behave differently.

Definition 2.6. (Observational preorder) Let M, N be processes. $M \sqsubseteq_{O} N$ if for all contexts $C[\,\cdot\,]$, for all names c,

$$C[M] \Downarrow_{\bar{c}}^{\mathrm{may}} \textit{ implies } C[N] \Downarrow_{\bar{c}}^{\mathrm{may}} .$$

Remark 2.7. In the definition above, we choose as behaviour of the terms, the $\Downarrow_{\bar{c}}^{\mathrm{may}}$ convergency. Choosing $\Downarrow_{c}^{\mathrm{may}}$ would be equivalent – in fact for all processes M, for all contexts $C[\,\cdot\,]$, for all names c, we have that

— $C[M] \Downarrow_{\bar{c}}^{\mathrm{may}}$ if and only if $\nu d\,(\nu c\,(C[M]|c(v).d(x).\mathbf{0})|\bar{d}y.c(z).\mathbf{0}) \Downarrow_{c}^{\mathrm{may}}$, and
— $C[M] \Downarrow_{c}^{\mathrm{may}}$ if and only if $\nu d\,(\nu c\,(C[M]|\bar{c}v.d(x).\mathbf{0})|\bar{d}y.\bar{c}z.\mathbf{0}) \Downarrow_{\bar{c}}^{\mathrm{may}}$,

where $d, v, x, y, z \notin fn(C[M])$.

Proposition 2.8. $M \sqsubseteq_{O} N$ implies $M \sqsubseteq_{OT} N$.

Proof. For every substitution on names s we can consider its restriction to $fn(M) \cup fn(N)$, say $s' = \{a_1/b_1, \ldots, a_n/b_n\}$ $(n \geqslant 1)$, and define a context

$$C_{s'}[\,\cdot\,] = \nu d\,(d(b_1).\cdots d(b_n).[\,\cdot\,]|\bar{d}a_1.\cdots \bar{d}a_n.E),$$

[†] Testers are usually called observers: we prefer testers to avoid confusion with the observational preorder.

where d is a fresh name. The context above when filled by M reduces to

$$\nu d(s(M)|E)$$

(and similarly for N). $\square$

The full abstraction result of Section 5 will show that the observational preorder and the open testing preorder are indeed the same preorder.

The second preorder is the testing preorder (Hennessy 1991; Boreale and De Nicola 1995).

Definition 2.9. (Testing preorder) Let M, N be processes. $M \sqsubseteq_T N$ if for every tester E,

$$M|E \Downarrow^{\text{may}}_{\bar{w}} \textit{ implies } N|E \Downarrow^{\text{may}}_{\bar{w}} .$$

It is clear that if $M \sqsubseteq_{OT} N$ then $M \sqsubseteq_T N$. The reverse implication does not hold in general, but from the full abstraction result of Section 5 we have that it holds for a subset of Π that includes the set of mono terms and the set of terms that do not contain matchings, | and !.

Remark 2.10. It is well known that the internal choice (the operator $\oplus$ in De Nicola and Hennessy (1987)) can be simulated in π-calculus. In fact, it is easy to verify that the process $M \equiv \nu x(\bar{x}y.P|\bar{x}y.Q|x(z).\mathbf{0})$ (where $x \notin fn(P) \cup fn(Q)$) and the internal choice between P and Q are strongly bisimilar. It is also easy to see that, with respect to 'may' preorders like the one considered in this paper, the internal choice and both the external choices (the operator [ ] in De Nicola and Hennessy (1987) and the operator + in Milner (1980)) have an identical behaviour. These facts motivate our choice of process constructors.

3. The set of types

In this section we introduce the types and show some of their properties.

In type assignment systems for polymorphic λ-calculus, types are properties of type free objects rather than domains in which objects live. This is common to many formalisms used in the theory of programming languages, and, in particular, to Hennessy–Milner logic for concurrency theory. Hennessy (1994a) gives a logic for processes that combines the functional arrow and an (implicit) intersection with indexed modalities.

We present our process logic as a standard type assignment system in which intersection appears explicitly. Instead of modalities, we use types to represent the interaction capabilities of a process. Moreover, we have type constructors corresponding to the channel binder ν and to the matching over channels.

Definition 3.1. The set $\mathscr{T}$ of *types*, ranged over by σ, τ and ρ, is generated by the following grammar:

$$\sigma ::= \omega \;\Big|\; \langle\alpha\rangle\sigma \;\Big|\; \sigma \wr \sigma \;\Big|\; \nu x\,\sigma \;\Big|\; [x = y]\sigma \;\Big|\; \sigma \wedge \sigma$$

where $\langle\alpha\rangle$ is called *prefix* and can be either an *input* or an *output*:

$$\alpha ::= x(y) \;\Big|\; \bar{x}y.$$

The type ω can be assigned to every process (it is the 'true' constant). A type $\langle\alpha\rangle\sigma$ is the type of a process that may offer the corresponding communication. For example, $\langle x(y)\rangle\sigma$ can be assigned to a process that may offer an input communication on channel x and that, after the communication, becomes a process having type σ, in which the name received is substituted for y. Parallel composition of processes is represented by the type constructor $\wr$. Finally, the $\wedge$ operator represents the logical 'and' of formulas. So, a process having type $\sigma_1 \wedge \sigma_2$ behaves as a process having both the types σ_1 and σ_2. Consider, for instance, the process M defined by $P|Q$, where $P \equiv x(y).\bar{y}y.\mathbf{0}$ and $Q \equiv \bar{x}z.\mathbf{0}$. The process M has type $\sigma \wr \tau$ where σ is a type of P and τ a type of Q. The process P offers the input communication $\langle x(y)\rangle$, and Q offers the output communication $\langle\bar{x}z\rangle$. Moreover, P and Q may communicate, yielding a process that offers the communication $\langle\bar{z}z\rangle$. So, among the types we can assign to M we have $\langle x(y)\rangle\omega$, $\langle\bar{x}z\rangle\omega$, and $\langle\bar{z}z\rangle\omega$. We can also have any intersection of such types.

The intersection operator does not correspond to any syntactic construct of the process language. But, as we will see in Remark 4.8, it corresponds to the internal choice operator $\oplus$ as defined in De Nicola and Hennessy (1987).

In the following we will use $\langle x[z = (y)]\rangle$ for $\langle x(y)\rangle[z = y]$, and $\langle\bar{x}(y)\rangle$ for $\nu y\langle\bar{x}y\rangle$. We call $\langle x[z = (y)]\rangle$ a matched input and $\langle\bar{x}(y)\rangle$ a bounded output. ζ will range over inputs, matched inputs, outputs and bounded outputs. Syntactic equality between types (modulo the above shortcuts) will be denoted by '='. Free and bound names of types are defined as for processes; $fn(\sigma)$ will denote the set of free names occurring in σ. The substitution of x for y in σ will be denoted by $\sigma\{x/y\}$. In writing types, we will assume the convention that substitution precedes channel binder, prefixing, and matching, which precede parallel composition $\wr$, which precedes intersection.

We note that there is a one–one correspondence between process constructors and type constructors with the exceptions of ! and $\wedge$.

Types are related by a preorder, which represents logical implication if they are thought of as properties, or, equivalently, subset inclusion if they are thought of as sets (the extensions of properties). Such a preorder induces an equivalence between types.

Type inclusion is intended to model the behaviour of processes roughly in the following sense. Let σ, τ, and ρ be the properties that characterize the processes M, P, and Q, respectively. If M is structurally congruent to P, then we assume that σ and τ are equivalent. Instead, if M in any context behaves either like P or like Q, we assume the equivalence of σ and $\tau\wedge\rho$. Other type equivalences are postulated, since there are processes that cannot be distinguished by any context. For example, if $x \neq y, z$, then $\nu x\, y(z).M$ and $y(z).\nu x\, M$ never exhibit a different behaviour. So we assume the equivalence between the types $\nu x\,\langle y(z)\rangle\sigma$ and $\langle y(z)\rangle\nu x\,\sigma$.

The *type inclusion relation* $\leqslant$, together with the induced equivalence relation $\simeq$, is defined as the least binary relation on $\mathscr{T}$ that satisfies the axioms and rules presented in the following.
Axioms are grouped according to their meanings, to facilitate their explanations. In the following Axiom **A2**(3) will stand for the third axiom of the group **A2**, and so on.

Structural equivalence axioms

A1 $\sigma \simeq \tau$ if σ is α-convertible to τ

A2 Abelian monoid laws for $\wr$:

$\sigma \wr \omega \simeq \sigma$

$\sigma \wr \tau \simeq \tau \wr \sigma$

$\sigma \wr (\tau \wr \rho) \simeq (\sigma \wr \tau) \wr \rho$

A3 $\nu x\, \nu y\, \sigma \simeq \nu y\, \nu x\, \sigma$

$\nu x\, \sigma \wr \tau \simeq \nu x\, (\sigma \wr \tau) \quad \text{if } x \notin fn(\tau)$

A4 $[x = x]\sigma \simeq \sigma$.

The set of 'structural equivalences' mirrors the congruence of terms, see Definition 2.2. Axiom **A2**(3) states the associativity of $\wr$, so we can omit parentheses in writing parallel compositions of types. Note that Axiom $\nu x\, \omega \simeq \omega$ is missing, since it can be derived from the 'may' inclusion axioms **A11**.

Behavioural equivalence axioms

A5 $\nu x\, \langle\alpha\rangle\sigma \simeq \langle\alpha\rangle\nu x\, \sigma \quad \text{if } x \notin fn(\alpha)$

$\nu x\, [y = z]\sigma \simeq [y = z]\nu x\, \sigma \quad \text{if } x \neq y \text{ and } x \neq z$

A6 $[x = y]\sigma \simeq [y = x]\sigma$

$[x = y][a = b]\sigma \simeq [a = b][x = y]\sigma$

$[x = y][y = z]\sigma \simeq [x = y][y = z][x = z]\sigma$

$[x = y]\sigma \simeq [x = y]\sigma\{x/y\}$

$[x = y](\sigma \wr \tau) \simeq [x = y]\sigma \wr [x = y]\tau$

A7 $\nu x\, \langle x(y)\rangle\sigma \simeq \omega$

$\nu x\, \langle \bar{x}y\rangle\sigma \simeq \omega$

$\nu x\, [x = y]\sigma \simeq \omega \quad \text{if } x \neq y.$

The previous equivalences establish properties that cannot be distinguished in any reasonable model of π-calculus. Namely, equivalences in **A5** reflect the fact that the ν binder can be moved into and out of prefixes and matchings (provided they do not mention the quantified name).

The equivalences in **A6** say that matchings induce an equivalence relation on names. Moreover, for the process $P \simeq [x = y]M$, the matching $[x = y]$ can be considered a guard. Whenever the process P is in a context in which x is equal to y, it behaves like M. So, in this context, M behaves also as the process M', where M' is obtained from M by substituting some of the x's by y's and *vice versa.* The Axiom **A6**(4) mimics this behaviour. In a context in which x is different from y (for instance, they could be bound by two different restriction operators) P is the inactive process.

Equivalences **A7** say that communications on local channels (bound by ν) cannot be offered, and matchings with local names cannot be satisfied.

'May' equivalence axioms

A8 $\overrightarrow{[a = b]}\langle\alpha_1\rangle\sigma \wr \overrightarrow{[c = d]}\langle\alpha_2\rangle\tau \simeq \overrightarrow{[a = b]}\langle\alpha_1\rangle(\sigma \wr \overrightarrow{[c = d]}\langle\alpha_2\rangle\tau) \wedge \overrightarrow{[c = d]}\langle\alpha_2\rangle(\overrightarrow{[a = b]}\langle\alpha_1\rangle\sigma \wr \tau)$

where either α_1 is $\bar{t}_1 z_1$ and α_2 is $\bar{t}_2 z_2$

or α_1 is $x_1(y_1)$ and α_2 is $x_2(y_2)$, $y_1 \notin fn(\overrightarrow{[c = d]}\langle\alpha_2\rangle\tau)$ and $y_2 \notin fn(\overrightarrow{[a = b]}\langle\alpha_1\rangle\sigma)$

$\overrightarrow{[a=b]}\langle\alpha_1\rangle\sigma \wr \overrightarrow{[c=d]}\langle\alpha_2\rangle\tau \ \simeq$
$\quad\overrightarrow{[a=b]}\langle\alpha_1\rangle(\sigma \wr \overrightarrow{[c=d]}\langle\alpha_2\rangle\tau) \wedge \overrightarrow{[c=d]}\langle\alpha_2\rangle(\overrightarrow{[a=b]}\langle\alpha_1\rangle\sigma \wr \tau) \wedge \overrightarrow{[a=b]}\overrightarrow{[c=d]}[x=t](\sigma\{z/y\} \wr \tau)$ †
$\quad$ where α_1 is $x(y)$, α_2 is $\bar{t}z$ and $y \notin fn(\overrightarrow{[c=d]}\langle\alpha_2\rangle\tau)$.

A9 $\langle a(b)\rangle[x=y]\sigma \simeq \langle a(b)\rangle\omega \wedge [x=y]\langle a(b)\rangle\sigma$ $\quad$ if $b \neq x$ and $b \neq y$
$\quad\langle \bar{a}b\rangle[x=y]\sigma \simeq \langle \bar{a}b\rangle\omega \wedge [x=y]\langle \bar{a}b\rangle\sigma$

A10 $\sigma \simeq \sigma \wedge \sigma$
$\quad\langle\alpha\rangle(\sigma\wedge\tau) \simeq \langle\alpha\rangle\sigma \wedge \langle\alpha\rangle\tau$
$\quad\rho \wr (\sigma\wedge\tau) \simeq \rho \wr \sigma \wedge \rho \wr \tau$
$\quad\nu x\,(\sigma\wedge\tau) \simeq \nu x\,\sigma \wedge \nu x\,\tau$
$\quad[x=y](\sigma\wedge\tau) \simeq [x=y]\sigma \wedge [x=y]\tau.$

The 'may equivalences' characterize the behaviour of parallel processes (note the similarity between the Axioms **A8** and the *expansion law* in Milner *et al.* (1992)). Consider, for instance, the equivalence **A8**(2). As already mentioned, this says that the behaviour of a parallel process is the 'and' of the behaviours of the individual processes and the result of their interaction (when the interaction is possible). Interaction is possible when we have a process offering a communication and a process expecting one. The fact that the input and output channels have to be the same is forced by the use of the matching operator. We can see also that the equivalence is given for types preceded by matchings. This is because matchings cannot be moved in or out of prefixes. Without matchings in the left-hand side, the equivalence **A8**(2) rewrites as

$$\langle x(y)\rangle\sigma \wr \langle\bar{t}z\rangle\tau \ \simeq\ \langle x(y)\rangle(\sigma \wr \langle\bar{t}z\rangle\tau) \wedge \langle\bar{t}z\rangle(\langle x(y)\rangle\sigma \wr \tau) \wedge [x=t](\sigma\{z/y\} \wr \tau).$$

There are also equivalences characterizing the logical operator $\wedge$. For a process M to have property $\sigma\wedge\tau$ means that M has both property σ and property τ. As mentioned earlier, $\wedge$ corresponds to the internal choice. The meaning of the equivalence **A9**(1) is that $a(b).[x=y]M$ certainly offers the input communication on a, so it has property $\langle a(b)\rangle\omega$. Moreover, if x is equal to y, it offers the input communication on a followed by M, so it has type $[x=y]\langle a(b)\rangle\sigma$ if σ characterizes a property of M. So it behaves like the internal choice between the processes $a(b).\mathbf{0}$ and $[x=y]a(b).M$. The equivalence **A9**(2) can be explained similarly. The equivalences in **A10** say that $\wedge$ is idempotent and it distributes over restrictions, matchings, prefixes and parallel compositions.

'May' inclusion axioms

A11 $\sigma \leqslant \omega$
$\quad\sigma\{y/x\} \leqslant \nu x\sigma$
$\quad\sigma \leqslant [x=y]\sigma$
$\quad\sigma\wedge\tau \leqslant \sigma$
$\quad\sigma\wedge\tau \leqslant \tau.$

The axioms for inclusion tell us when a process behaves better than another in a 'may' perspective. First, every process offers more communications than the one that offers none. The second axiom means that restricting a process by ν produces a process that offers

† Here and in the following we omit parentheses since, as proved in Remark 3.2, $\wedge$ is associative.

fewer communications. The reason for having $\sigma\{y/x\}$ instead of σ is that this is true also when the ν captures only some of the occurrences of y. For example $\langle\bar{y}y\rangle\omega \leqslant \nu x\langle\bar{y}x\rangle\omega$ by choosing σ to be $\langle\bar{y}x\rangle\omega$. Also, restricting a process by a matching produces a worse process (third axiom). We also have the basic logical property of $\wedge$. Note that from these inclusions we immediately derive $\omega \simeq \nu x\omega \simeq [x = y]\omega$.

Rules

R1 $\sigma \leqslant \sigma'$ implies $\langle\alpha\rangle\sigma \leqslant \langle\alpha\rangle\sigma'$
$\sigma \leqslant \sigma'$ implies $\nu x\sigma \leqslant \nu x\sigma'$
$\sigma \leqslant \sigma'$ implies $[x = y]\sigma \leqslant [x = y]\sigma'$
$\sigma \leqslant \sigma'$ and $\tau \leqslant \tau'$ imply $\sigma \wr \tau \leqslant \sigma' \wr \tau'$
$\sigma \leqslant \sigma'$ and $\tau \leqslant \tau'$ imply $\sigma \wedge \tau \leqslant \sigma' \wedge \tau'$
R2 $\sigma \leqslant \tau$ and $\tau \leqslant \rho$ imply $\sigma \leqslant \rho$.

The rules extend inclusions (and so also equivalences) to the various syntactic forms.

Remark 3.2. We can prove that $\wedge$ is associative, that is, $\sigma \wedge (\tau \wedge \rho) \simeq (\sigma \wedge \tau) \wedge \rho$. We first show that $\sigma \wedge (\tau \wedge \rho) \leqslant (\sigma \wedge \tau) \wedge \rho$.

1 $\sigma \wedge (\tau \wedge \rho) \leqslant \sigma$ using **A11**(4),
2 $\sigma \wedge (\tau \wedge \rho) \leqslant \tau \wedge \rho$ from **A11**(4), and $\tau \wedge \rho \leqslant \tau$ again from **A11**(4), and $\tau \wedge \rho \leqslant \rho$ using **A11**(5). Therefore from Rule **R2** we have
(a) $\sigma \wedge (\tau \wedge \rho) \leqslant \tau$, and
(b) $\sigma \wedge (\tau \wedge \rho) \leqslant \rho$.
3 From Points 1, and 2a, we derive that $\sigma \wedge (\tau \wedge \rho) \leqslant \sigma \wedge \tau$, using **A10**(1) and **R1**(5). Finally from this, and 2b, again using **A10**(1) and **R1**(5), we get $\sigma \wedge (\tau \wedge \rho) \leqslant (\sigma \wedge \tau) \wedge \rho$.

The other inclusion $(\sigma \wedge \tau) \wedge \rho \leqslant \sigma \wedge (\tau \wedge \rho)$ can be proved in a similar way.

It is not difficult (but quite cumbersome) to verify that $\leqslant$ is a decidable relation. We do not do this, since it is not relevant for our development.

It is easy to see that $\leqslant$ is a precongruence and that $\simeq$ is a congruence over types. Moreover, $\sigma \leqslant \tau$ implies that $\sigma\{y/x\} \leqslant \tau\{y/x\}$.

Because of Axioms **A8**, we have that any equivalence class of $\simeq$ contains a type in which the parallel composition $\wr$ does not occur. Indeed, the behaviour of a parallel process can be given in terms of the intersection of the behaviours of its components. Moreover, Axioms **A10** say that the $\wedge$ operator can always be pulled out of the various syntactic forms, that is, any type is equivalent to an intersection of types that do not contain any occurrence of $\wr$ and $\wedge$. Looking at Axioms **A7**, we can see that subtypes of types can be replaced by ω when they correspond to behaviours not offering any communication. Moreover, Axioms **A4** and **A6** allow us to erase superfluous matchings and to choose a representative for each induced equivalence class of names. So, types can be shortened to obtain equivalent types. The fourth axiom of **A6** allows us to have types in which the names occurring on the right of $=$ in a matching do not occur anywhere else. Finally, Axioms **A5** allow us to move the ν binder into and out of prefixes and matchings that do not involve the bound variable. These considerations lead us to formulate the notions of bodies and normal types.

Definition 3.3. (Normal types)

1 A *normal sequence of matchings* is inductively defined by

— the empty sequence of matchings is normal,

— $[x = y]\overrightarrow{[a = b]}$ is a normal sequence of matchings, if and only if $\overrightarrow{[a = b]}$ is normal, $x \neq y$, $y \notin \overrightarrow{[a = b]}$ and $x \notin \vec{b}$.

2 The set of *bodies* $\mathscr{B}$ is inductively defined by

— $\omega \in \mathscr{B}$

— $\langle x(y)\rangle\beta, \langle\bar{x}y\rangle\beta \in \mathscr{B}$ if $\beta \in \mathscr{B}$

— $\langle x[z = (y)]\rangle\beta \in \mathscr{B}$ if $z \neq y$, $\beta \in \mathscr{B}$ and $y \notin fn(\beta)$

— $\langle\bar{x}(y)\rangle\beta \in \mathscr{B}$ if $x \neq y$ and $\beta \in \mathscr{B}$.

3 A type ϕ is *normal* if and only if

— either $\phi = \omega$,

— or $\phi = \overrightarrow{[a = b]}\beta$, where $\overrightarrow{[a = b]}$ is a normal sequence of matchings, $\beta \neq \omega$ is a body and $\vec{b} \notin fn(\beta)$.

A sequence of matchings induces an equivalence relation on names: $\mathscr{E}(\overrightarrow{[a = b]})$ will denote the equivalence relation induced by $\overrightarrow{[a = b]}$. A normal sequence is a standard representation of an equivalence relation. Notice that a normal sequence is always a minimal representation of an equivalence relation in the sense that it contains a minimal number of matchings. When the matching $[x = y]$ belongs to a normal sequence, we say that x is the *representative* of the equivalence class of y.

Proposition 3.4.

1 Let $\overrightarrow{[a = b]}$ and $\overrightarrow{[x = y]}$ be sequences of matchings, and let σ be a type. $\mathscr{E}(\overrightarrow{[a = b]}) = \mathscr{E}(\overrightarrow{[x = y]})$ implies $\overrightarrow{[a = b]}\sigma \simeq \overrightarrow{[x = y]}\sigma$.

2 If $\overrightarrow{[x = y]}$ is a sequence of matchings, then there is a normal sequence $\overrightarrow{[a = b]}$ such that $\mathscr{E}(\overrightarrow{[x = y]}) = \mathscr{E}(\overrightarrow{[a = b]})$.

Proof.

1 Observe that $\mathscr{E}(\overrightarrow{[a = b]}) = \mathscr{E}(\overrightarrow{[x = y]})$ and $[z = t] \in \overrightarrow{[x = y]}$ imply that there are $c_0, \dots, c_n$ $(n \geqslant 1)$ such that:

— $c_0 = z$;

— $[c_i = c_{i+1}]$ or $[c_{i+1} = c_i] \in \overrightarrow{[a = b]}$;

— $c_n = t$.

Therefore we get $\overrightarrow{[a = b]}\sigma \simeq \overrightarrow{[a = b]}\,\overrightarrow{[x = y]}\sigma$ using the first three equivalences in **A6**. Symmetrically, we have $\overrightarrow{[x = y]}\sigma \simeq \overrightarrow{[x = y]}\,\overrightarrow{[a = b]}\sigma$. Finally, using **A6**(2), we get $\overrightarrow{[a = b]}\,\overrightarrow{[x = y]}\sigma \simeq \overrightarrow{[x = y]}\,\overrightarrow{[a = b]}\sigma$. Therefore, by transitivity of $\simeq$ (Rule **R2**), $\overrightarrow{[a = b]}\sigma \simeq \overrightarrow{[x = y]}\sigma$.

2 We use structural induction on $\overrightarrow{[x = y]}$. If $\overrightarrow{[x = y]} = [z = t]\overrightarrow{[u = v]}$, then by induction there is a normal sequence of matchings $\overrightarrow{[c = d]}$ such that $\mathscr{E}(\overrightarrow{[u = v]}) = \mathscr{E}(\overrightarrow{[c = d]})$. Now, if $z = t \in \mathscr{E}(\overrightarrow{[u = v]})$ (In particular, we can have z identical to t), then we choose

$\overrightarrow{[a=b]} = \overrightarrow{[c=d]}$. Otherwise, if $z \notin \vec{d}$ and $t \notin \overrightarrow{[c=d]}$, then we choose $\overrightarrow{[a=b]} = [z = t]\overrightarrow{[c=d]}$. If $z,t \in \vec{d}$, let e,f be the representative of the corresponding equivalence classes. $z = t \notin \mathscr{E}(\overrightarrow{[u=v]})$ implies $e \neq f$. We choose $\overrightarrow{[a=b]} = [e=f](\overrightarrow{[c=d]}\{e/f\})$. If $z,t \in \vec{c}$, we choose $\overrightarrow{[a=b]} = [z=t](\overrightarrow{[c=d]}\{z/t\})$. If $z \in \vec{c}$ and $t \in \vec{d}$, we choose $\overrightarrow{[a=b]} = [z=e](\overrightarrow{[c=d]}\{z/e\})$, where e is the representative of the equivalence class of t. The remaining cases have z and t interchanged, so we can do similar choices. □

Observe that normal types represent traces of communications that may be offered by a process, namely a match specifies a condition on names (that is, on the context in which the process is embedded) that is necessary for the process to satisfy the specification that follows the match. A prefix specifies when the corresponding communication may be offered. Moreover, a prefix $\langle\bar{x}(y)\rangle\beta$ specifies that a new name is created, and a prefix $\langle x[z=(y)]\rangle\beta$ specifies a condition on the received name necessary for the process to satisfy the specification that follows the prefix. The results that follow will show that every type can be expressed as an intersection of normal types. This is consistent with the intuition that the behaviour of a process is determined by all the possible traces of communications that it may offer. Note that, in spite of this, our semantics is not a trace semantics, since we compare types by means of an inclusion relation that takes into account the behaviour of processes in contexts.

In the following, we give a definition of substitutions (normal substitutions) and v bindings (normal bindings) mapping normal types into normal types. Note that, if β is a body, then $\beta\{x/y\}$ is a body, but ϕ normal does not imply $\phi\{x/y\}$ normal. For example, if ϕ is $[x=y]\langle\alpha\rangle\omega$, then $\phi\{x/y\} = [x=x]\langle\alpha\rangle\omega$, which is not normal. Moreover, $vx\beta$ is not in general normal either when β is a body. In order to define normal substitutions, we need to force a given name, say x, to be the representative of the equivalence class (of the equivalence relation) induced by a sequence of matchings. This is done by the following mapping $\mathsf{rep}(x,\phi)$.

Let $\overrightarrow{[a=b]} - [x=y]$ denote the sequence of matchings obtained from $\overrightarrow{[a=b]}$ by erasing $[x=y]$, if it occurs.

Definition 3.5. Let $\phi = \overrightarrow{[a=b]}\beta$ be a normal type, where β is a body.

1

$$\mathsf{rep}(x,\phi) = \begin{cases} [x=c]((\overrightarrow{[a=b]} - [c=x])\beta)\{x/c\} & \text{if } [c=x] \in \overrightarrow{[a=b]} \\ \phi & \text{otherwise.} \end{cases}$$

2 To define the normal substitution of x for y in ϕ, first let us define $\psi\{x/y\}^{\mathsf{R}}$. In the definition of $\psi\{x/y\}^{\mathsf{R}}$, we assume that both x and y are the representative of their equivalence classes in ψ if their equivalence classes are different, otherwise x is the representative of the common equivalence class. $\psi\{x/y\}^{\mathsf{R}}$ is inductively defined as follows:

— $\beta\{x/y\}^{\mathsf{R}} = \beta\{x/y\}$,

— $([x=y]\psi')\{x/y\}^{\mathsf{R}} = \psi'$,

— $([z=a]\psi')\{x/y\}^{\mathsf{R}} = [x=a](\psi'\{x/y\}^{\mathsf{R}})$, for $z=x$ or $z=y$, and $a \neq x,y$,

— $([a=b]\psi')\{x/y\}^{\mathsf{R}} = [a=b](\psi'\{x/y\}^{\mathsf{R}})$, for $a \neq x,y$ and $b \neq x,y$.

Now define the *normal substitution* of x for y in ϕ, $\phi\{x/y\}^{\mathsf{N}}$, to be

$$(\mathsf{rep}(x, \mathsf{rep}(y, \phi)))\{x/y\}^{\mathsf{R}}.$$

3 The *normal binding* of x in ϕ, $\nu^{\mathsf{N}} x\,\phi$, is inductively defined as follows:

— $\nu^{\mathsf{N}} x\,\omega = \omega$,

— $\nu^{\mathsf{N}} x\,\langle\zeta\rangle\beta = \langle\zeta\rangle\nu^{\mathsf{N}} x\,\beta$, for $x \notin fn(\zeta)$,

— $\nu^{\mathsf{N}} x\,\langle\zeta\rangle\beta = \omega$, for $\zeta \in \{x(y), \bar{x}y, x[z = (y)], \bar{x}(y)\}$,

— $\nu^{\mathsf{N}} x\,\langle\bar{y}x\rangle\beta = \langle\bar{y}(x)\rangle\beta$, for $x \neq y$,

— $\nu^{\mathsf{N}} x\,\langle y[x = (z)]\rangle\beta = \langle y(z)\rangle\omega$, for $x \neq y$,

— $\nu^{\mathsf{N}} x\,\overrightarrow{[a = b]}\beta = \begin{cases} \omega & \text{for } x \in \overrightarrow{[a = b]} \text{ or } \nu^{\mathsf{N}} x\,\beta = \omega \\ \overrightarrow{[a = b]}\nu^{\mathsf{N}} x\,\beta & \text{otherwise.} \end{cases}$

The definition of normal substitution is justified by observing that in $\mathsf{rep}(x, \mathsf{rep}(y, \phi))$, both x and y are the representative of their equivalence classes in ϕ if x and y are in different equivalence classes, and x is the representative when they both belong to the same class.

Normal substitutions and normal bindings behave, respectively, like standard substitutions and standard bindings modulo type equivalence.

Definition 3.5 allows us to state some equivalences between types and intersections of normal types that can be shown easily and will be useful in the following.

Lemma 3.6. Let $\phi = \overrightarrow{[a = b]}\beta$ be normal, where β is a body. Then the following equivalences hold and each right-hand side is an intersection of normal types whose free names are a subset of the free names of the corresponding left-hand side.

1 $\phi \simeq \mathsf{rep}(x, \phi)$.

2 $\phi\{x/y\} \simeq \phi\{x/y\}^{\mathsf{N}}$.

3 $\langle x(y)\rangle\phi \simeq \begin{cases} \langle x(y)\rangle\omega \wedge \overrightarrow{[a = b]}\langle x\{\vec{a}/\vec{b}\}(y)\rangle\beta & \text{if } y \notin \overrightarrow{[a = b]} \\ \langle x(y)\rangle\omega \wedge \overrightarrow{[c = d]}\langle x\{\vec{c}/\vec{d}\}[z = (y)]\rangle(\beta\{z/y\}) & \text{if } [z = y] \text{ or} \\ & [y = z] \in \overrightarrow{[a = b]} \end{cases}$

where $\overrightarrow{[c = d]} = (\overrightarrow{[a = b]} - [z = y] - [y = z])\{z/y\}$.

4 $\langle\bar{x}y\rangle\phi \simeq \langle\bar{x}y\rangle\omega \wedge \overrightarrow{[a = b]}\langle\bar{x}y\rangle\{\vec{a}/\vec{b}\}\beta$.

5 $\nu x\,\phi \simeq \nu^{\mathsf{N}} x\,\phi$.

6 $[x = y]\phi \simeq \begin{cases} \omega & \text{if } \phi = \omega \\ [x = y](\phi\{x/y\}^{\mathsf{N}}) & \text{otherwise.} \end{cases}$

Proof. We only show Point 3. If $y \notin \overrightarrow{[a = b]}$, then

$$\langle x(y)\rangle\overrightarrow{[a = b]}\beta \simeq \langle x(y)\rangle\omega \wedge \overrightarrow{[a = b]}\langle x(y)\rangle\beta$$

by repeated applications of Axiom **A9**(1). Moreover,

$$\overrightarrow{[a = b]}\langle x(y)\rangle\beta \simeq \overrightarrow{[a = b]}(\langle x(y)\rangle\beta\{\vec{a}/\vec{b}\})$$

by Axiom **A6**(4), and

$$\langle x(y)\rangle\beta\{\vec{a}/\vec{b}\} = \overrightarrow{[a = b]}\langle x\{\vec{a}/\vec{b}\}(y)\rangle\beta,$$

since by definition of normal type, $\vec{b} \notin fn(\beta)$.

If $[z = y] \in \overrightarrow{[a = b]}$, let $\overrightarrow{[c = d]} = \overrightarrow{[a = b]} - [z = y]$. By **A6**(2) and **A9**(1), we get

$$\langle x(y)\rangle \overrightarrow{[a = b]}\beta \simeq \langle x(y)\rangle\omega \wedge \overrightarrow{[c = d]}\langle x(y)\rangle[z = y]\beta.$$

By notational convention, $\langle x(y)\rangle[z = y]\beta = \langle x[z = (y)]\rangle\beta$, so we conclude, as in previous case, $\overrightarrow{[c = d]}\langle x[z = (y)]\rangle\beta \simeq \overrightarrow{[c = d]}\langle x\{\vec{c}/\vec{d}\}[z = (y)]\rangle\beta$, since $\vec{d}, y \notin fn(\beta)$. Notice that in this case $y \notin \overrightarrow{[c = d]}$.

If $[y = z] \in \overrightarrow{[a = b]}$, Proposition 3.4(1) and Axiom **A6**(4) imply $\langle x(y)\rangle \overrightarrow{[a = b]}\beta \simeq \langle x(y)\rangle((\overrightarrow{[a = b]} - [y = z])\{z/y\})[z = y](\beta\{z/y\})$, and therefore, as in previous cases, $\langle x(y)\rangle((\overrightarrow{[a = b]} - [y = z])\{z/y\})[z = y](\beta\{z/y\}) \simeq \overrightarrow{[c = d]}\langle x\{\vec{c}/\vec{d}\}[z = (y)]\rangle(\beta\{z/y\})$, where $\overrightarrow{[c = d]} = (\overrightarrow{[a = b]} - [y = z])\{z/y\}$. □

We can now prove that every type can be expressed as a finite intersection of normal types.

Proposition 3.7. For every type σ there is $n \geqslant 1$ and there are n normal types $\phi_1, \ldots, \phi_n$ such that

$$\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_n \quad \textit{and} \quad fn(\sigma) \supseteq fn(\phi_1 \wedge \cdots \wedge \phi_n).$$

Proof. We use induction on σ. All the cases except for $\wr$ follow from the induction hypothesis and Lemma 3.6.
For $\wr$, let σ be $\tau \wr \tau'$. By the induction hypothesis, $\tau \simeq \psi_1 \wedge \cdots \wedge \psi_m$, and $\tau' \simeq \psi'_1 \wedge \cdots \wedge \psi'_n$, where $\psi_1, \cdots, \psi_m, \psi'_1, \cdots, \psi'_n$ are normal types. Then

$$\tau \wr \tau' \simeq \bigwedge_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n} \psi_i \wr \psi'_j.$$

So we have to show that for all normal types ψ and ψ', $\psi \wr \psi'$ is equivalent to an intersection of normal types. The proof is by induction on the number of prefixes in ψ and ψ'. If either $\psi \simeq \omega$ or $\psi' \simeq \omega$, the result is obvious. Otherwise we consider the most difficult case: $\psi = \overrightarrow{[a = b]}\langle x(y)\rangle\beta$ and $\psi' = \overrightarrow{[c = d]}\langle\bar{v}(t)\rangle\gamma$. We have:

$$\psi \wr \psi' \simeq \overrightarrow{[a = b]}\langle x(y)\rangle(\beta \wr \psi') \wedge \overrightarrow{[c = d]}\langle\bar{v}(t)\rangle(\psi \wr \gamma) \wedge \overrightarrow{[a = b]}\,\overrightarrow{[c = d]}[x = v]vt\,(\beta\{t/y\} \wr \gamma).$$

As already noted, $\beta\{t/y\}$ is a body, and therefore it is also a normal type. So applying the induction hypothesis to the three parallel compositions ($\beta \wr \psi'$, $\psi \wr \gamma$, and $\beta\{t/y\} \wr \gamma$), the equivalences **A10** (that pull out the $\wedge$ operator) and Lemma 3.6 (to add the matchings and the restrictions), we get the result. □

Remark 3.8. When we look at the previous proposition, a natural question arises. Why didn't we just consider normal types? This is, as will be clear from the following section, because the definition of the type assignment system and the proofs of its properties would be much more difficult when restricted to normal types.

In the rest of the present section we will relate the preorder on types to the decomposition in normal types (Theorem 3.14). To this end, it is useful to characterize in a structural way the inclusion relation between normal types.

Definition 3.9. (Inclusion relation for normal types)

1 The *normal inclusion relation*, $\leqslant^{\mathsf{N}}$, between normal types is inductively defined by:

— $\phi \leqslant^{\mathsf{N}} \omega$

— $\phi \leqslant^{\mathsf{N}} \psi$, if ϕ is α-convertible to ψ

— $\langle x(y)\rangle\beta \leqslant^{\mathsf{N}} \langle x[z = (y)]\rangle\beta\{z/y\}$, for $z \neq y$

— $\langle \bar{x}z\rangle\beta\{z/y\} \leqslant^{\mathsf{N}} \langle \bar{x}(y)\rangle\beta$, for $y \neq x$

— $\beta, \gamma \in \mathscr{B}$ and $\beta \leqslant^{\mathsf{N}} \gamma$ imply

- $\langle x(y)\rangle\beta \leqslant^{\mathsf{N}} \langle x(y)\rangle\gamma$
- $\langle \bar{x}y\rangle\beta \leqslant^{\mathsf{N}} \langle \bar{x}y\rangle\gamma$
- $\langle x[y = (z)]\rangle\beta\{z/y\} \leqslant^{\mathsf{N}} \langle x[y = (z)]\rangle\gamma\{z/y\}$, for $z \neq y$
- $\langle \bar{x}(y)\rangle\beta \leqslant^{\mathsf{N}} \langle \bar{x}(y)\rangle\gamma$, for $y \neq x$

— $\phi \leqslant^{\mathsf{N}} [x = y]\phi\{x/y\}^{\mathsf{N}}$

— $\phi \leqslant^{\mathsf{N}} \psi$ imply $[x = y]\phi\{x/y\}^{\mathsf{N}} \leqslant^{\mathsf{N}} [x = y]\psi\{x/y\}^{\mathsf{N}}$

— $\phi_1 \leqslant^{\mathsf{N}} \phi_2$, and $\phi_2 \leqslant^{\mathsf{N}} \phi_3$ imply $\phi_1 \leqslant^{\mathsf{N}} \phi_3$.

2 $\simeq^{\mathsf{N}}$ is the equivalence relation induced by $\leqslant^{\mathsf{N}}$.

From the previous definition it follows that two $\simeq^{\mathsf{N}}$ equivalent normal types can only differ for α-conversion, the order of the initial matchings and the choice of the representative of the equivalence classes in the initial sequences of matchings.

To relate the type inclusion, $\leqslant$, and the normal type inclusion, $\leqslant^{\mathsf{N}}$, between normal types, we define for any normal type ϕ a context $\Gamma_{\phi,w}[\,\cdot\,]$, where w is a fresh name, that discriminates types less than or equal to ϕ. First we associate to any body β a test body $\Im_{\beta,w}$ that offers the communications 'complementary' to those offered by β. Then if $\phi = \overrightarrow{[a = b]}\beta$, the context $\Gamma_{\phi,w}[\,\cdot\,]$ first satisfies the matchings $\overrightarrow{[a = b]}$ and then puts the hole in parallel with $\Im_{\beta,w}$.

Definition 3.10.

1 Let β be a body, and $w \notin fn(\beta)$. The *test body* $\Im_{\beta,w}$ for β is defined by induction on β.

— $\Im_{\omega,w} = \langle \bar{w}(v)\rangle\omega$

— $\Im_{\langle x(y)\rangle\gamma,w} = \langle \bar{x}(y)\rangle\Im_{\gamma,w}$

— $\Im_{\langle x[z=(y)]\rangle\gamma,w} = \langle \bar{x}z\rangle\Im_{\gamma,w}$

— $\Im_{\langle \bar{x}y\rangle\gamma,w} = \langle x[y = (z)]\rangle\Im_{\gamma,w}$

— $\Im_{\langle \bar{x}(y)\rangle\gamma,w} = \langle x(y)\rangle\Im_{\gamma,w}$.

2 Let $\phi = \overrightarrow{[a = b]}\beta$ and $w \notin fn(\phi)$. Define $\Gamma_{\phi,w}[\,\cdot\,]$ as

$$\nu d\,(\langle d(b_1)\rangle \cdots \langle d(b_n)\rangle [\,\cdot\,] \wr \langle \bar{d}a_1\rangle \cdots \langle \bar{d}a_n\rangle\Im_{\beta,w})$$

where $d \neq w$ and $d \notin fn(\phi)$.

Lemma 3.11. Let $\phi = \overrightarrow{[a = b]}\beta$ and ψ be normal types, $\vec{c} \supseteq fn(\phi) \cup fn(\psi)$ and $w \notin \vec{c}$.

1 $\overrightarrow{\nu c}\,\Gamma_{\phi,w}[\psi] \not\simeq \omega$ implies $\psi \leqslant^{\mathsf{N}} \phi$, and

2 $\psi \leqslant \phi$ implies $\overrightarrow{\nu c}\,\Gamma_{\phi,w}[\psi] \leqslant \langle \bar{w}(v)\rangle\omega$.

Proof.

1 We use induction on β.

If $\beta = \omega$, then $\phi \simeq \omega$, so it is trivial.

Let $\beta \neq \omega$. Note that Axioms **A8**(2), **A7**(1), and **A7**(2) imply

$$\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \simeq \overrightarrow{\nu c}\nu d(\psi\{\vec{a}/\vec{b}\} \wr \Im_{\beta,w}). \qquad (*)$$

First observe that $\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \not\simeq \omega$ implies that $\psi\{\vec{a}/\vec{b}\}$ is a body. This is because if $\psi\{\vec{a}/\vec{b}\} = [e = f]\psi'$ and $\Im_{\beta,w} = \langle\zeta\rangle\beta'$, then

$$\begin{array}{lll} \overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] & \simeq \overrightarrow{\nu c}\nu d([e = f]\psi' \wr \langle\zeta\rangle\beta') & \text{by } (*) \\ & \simeq \overrightarrow{\nu c}\nu d[e = f]\sigma_1 \wedge \overrightarrow{\nu c}\nu d\langle\zeta\rangle\sigma_2 \text{ for some } \sigma_1 \text{ and } \sigma_2 & \text{by Axioms } \mathbf{A8} \\ & \simeq \omega & \text{by Axioms } \mathbf{A7} \end{array}$$

since e, f, and the free names in ζ are all included in $\vec{c}$.

Let $\beta = \langle x(y)\rangle\gamma$, then $\Im_{\beta,w} = \langle\bar{x}(y)\rangle\Im_{\gamma,w}$ and

$$\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \simeq \overrightarrow{\nu c}\nu d(\psi\{\vec{a}/\vec{b}\} \wr \langle\bar{x}(y)\rangle\Im_{\gamma,w}) \text{ by } (*).$$

For $\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \not\simeq \omega$, it must be that $\psi\{\vec{a}/\vec{b}\} = \langle x(z)\rangle\delta$ for some δ such that

$$\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \simeq \overrightarrow{\nu c}\nu d(\delta\{y/z\} \wr \Im_{\gamma,w}) = \overrightarrow{\nu c}\Gamma_{\gamma,w}\left[\delta\{y/z\}\right] \not\simeq \omega.$$

By the induction hypothesis, $\delta\{y/z\} \leqslant^{\mathsf{N}} \gamma$.

So $\psi\{\vec{a}/\vec{b}\} = \langle x(z)\rangle\delta \simeq^{\mathsf{N}} \langle x(y)\rangle\delta\{y/z\} \leqslant^{\mathsf{N}} \langle x(y)\rangle\gamma = \beta$ using Definition 3.9. Since $\psi \leqslant^{\mathsf{N}} \overrightarrow{[a = b]}\psi\{\vec{a}/\vec{b}\}$, we have $\psi \leqslant^{\mathsf{N}} \overrightarrow{[a = b]}\beta$ by the same definition.

Let $\beta = \langle x[z = (y)]\rangle\gamma$, then $\Im_{\beta,w} = \langle\bar{x}z\rangle\Im_{\gamma,w}$. For $\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \not\simeq \omega$ it must be that either $\psi\{\vec{a}/\vec{b}\} = \langle x(t)\rangle\delta$ or $\psi\{\vec{a}/\vec{b}\} = \langle x[z = (t)]\rangle\delta$ for some δ. In both cases $\psi\{\vec{a}/\vec{b}\} \leqslant^{\mathsf{N}} \langle x[z = (t)]\rangle\delta$ and

$$\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \simeq \overrightarrow{\nu c}\nu d(\delta\{z/t\} \wr \Im_{\gamma,w}) = \overrightarrow{\nu c}\Gamma_{\gamma,w}\left[\delta\{z/t\}\right] \not\simeq \omega.$$

By the induction hypothesis, $\delta\{z/t\} \leqslant^{\mathsf{N}} \gamma$. So $\langle x[z = (t)]\rangle\delta\{z/t\} \leqslant^{\mathsf{N}} \langle x[z = (y)]\rangle\gamma$ (since $y \notin fn(\gamma)$ by definition of normal type). As for the previous case, we can now deduce that $\psi \leqslant^{\mathsf{N}} \overrightarrow{[a = b]}\beta$.

Let $\beta = \langle\bar{x}y\rangle\gamma$, $\Im_{\beta,w} = \langle x[y = (z)]\rangle\Im_{\gamma,w}$. For $\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \not\simeq \omega$, it must be that $\psi\{\vec{a}/\vec{b}\} = \langle\bar{x}y\rangle\delta$ for some δ. Indeed, if $\psi\{\vec{a}/\vec{b}\}$ were $\langle\bar{x}t\rangle\delta$, or $\langle\bar{x}(t)\rangle\delta$, then the matching $[y = z]$ in $\Im_{\beta,w}$ after the communication would become $[y = t]$ with t bound by an external ν, and therefore $\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right]$ would be ω. So we have that

$$\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \simeq \overrightarrow{\nu c}\nu d(\delta \wr \Im_{\gamma,w}) = \overrightarrow{\nu c}\Gamma_{\gamma,w}\left[\delta\right] \not\simeq \omega.$$

By the induction hypothesis, $\delta \leqslant^{\mathsf{N}} \gamma$. So $\langle\bar{x}y\rangle\delta \leqslant^{\mathsf{N}} \langle\bar{x}y\rangle\gamma$, and, as for the previous cases, we can conclude that $\psi \leqslant^{\mathsf{N}} \overrightarrow{[a = b]}\beta$.

Let $\beta = \langle\bar{x}(y)\rangle\gamma$, $\Im_{\beta,w} = \langle x(y)\rangle\Im_{\gamma,w}$. For $\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \not\simeq \omega$ it must be that either $\psi\{\vec{a}/\vec{b}\} = \langle\bar{x}z\rangle\delta$ or $\psi\{\vec{a}/\vec{b}\} = \langle\bar{x}(z)\rangle\delta$ for some δ. In both cases

$$\overrightarrow{\nu c}\Gamma_{\phi,w}\left[\psi\right] \simeq \overrightarrow{\nu c}\ \nu z^{\dagger}\ \nu d(\delta \wr \Im_{\gamma,w}\{z/y\}) = \overrightarrow{\nu c}\ \nu z\Gamma_{\gamma\{z/y\},w}\left[\delta\right] \not\simeq \omega,$$

† If $\psi\{\vec{a}/\vec{b}\} = \langle\bar{x}z\rangle\delta$ we get $z \in \vec{c}$, so this binding is superfluous.

since $\Im_{\gamma,w}\{z/y\} = \Im_{\gamma\{z/y\},w}$. By the induction hypothesis, we derive $\delta \leqslant^{\mathsf{N}} \gamma\{z/y\}$.
If $\psi\{\vec{a}/\vec{b}\} = \langle \bar{x}(z)\rangle\delta$, observe that $\langle \bar{x}(z)\rangle\delta \leqslant^{\mathsf{N}} \langle \bar{x}(z)\rangle\gamma\{z/y\}$. Since $\langle \bar{x}(z)\rangle\gamma\{z/y\}$ is α-convertible to β under the condition $z \notin fn(\gamma)$, we can derive the result.
If $\psi\{\vec{a}/\vec{b}\} = \langle \bar{x}z\rangle\delta$, observe that $\langle \bar{x}z\rangle\delta \leqslant^{\mathsf{N}} \langle \bar{x}z\rangle\gamma\{z/y\} \leqslant^{\mathsf{N}} \langle \bar{x}(y)\rangle\gamma = \beta$, and we can derive the result.

2 Let $\psi \leqslant \phi$, then, since $\leqslant$ is a precongruence, we have that

$$\overline{v\vec{c}}\Gamma_{\phi,w}\left[\psi\right] \leqslant \overline{v\vec{c}}\Gamma_{\phi,w}\left[\phi\right].$$

Moreover, $\overline{v\vec{c}}\Gamma_{\phi,w}\left[\phi\right] \simeq \overline{v\vec{c}}vd(\beta \wr \Im_{\beta,w})$ by (∗), since $\beta\{\vec{a}/\vec{b}\} = \beta$. It is easy to see that $\beta \wr \Im_{\beta,w} \leqslant \langle \bar{w}(v)\rangle\omega$. So $\overline{v\vec{c}}\Gamma_{\phi,w}\left[\phi\right] \leqslant \langle \bar{w}(v)\rangle\omega$, and therefore $\overline{v\vec{c}}\Gamma_{\phi,w}\left[\psi\right] \leqslant \langle \bar{w}(v)\rangle\omega$. □

From the previous lemma we get the coincidence of $\leqslant$ and $\leqslant^{\mathsf{N}}$ on normal types.

Corollary 3.12. Let ϕ and ψ be normal types. $\phi \leqslant \psi$ if and only if $\phi \leqslant^{\mathsf{N}} \psi$.

Proof. The 'if' implication is by induction on the definition of $\leqslant^{\mathsf{N}}$ and the 'Only If' implication follows directly from Lemma 3.11. □

Define τ to be a *prime* type if and only if for all σ and σ', if $\sigma \wedge \sigma' \leqslant \tau$, then either $\sigma \leqslant \tau$, or $\sigma' \leqslant \tau$. The following theorem shows that the set of normal types coincides with the set of prime types (modulo equivalence).

Theorem 3.13.

1 Any normal type is prime.
2 Any prime type is equivalent to a normal type.

Proof.

1 Let ϕ be a normal type, and σ and σ' be types. By Proposition 3.7, $\sigma \simeq \psi_1 \wedge \cdots \wedge \psi_m$ and $\sigma' \simeq \psi'_1 \wedge \cdots \wedge \psi'_n$, where $\psi_1, \cdots, \psi_m, \psi'_1, \cdots, \psi'_n$ are normal types. Let $\vec{c} = fn(\bigwedge_{1\leqslant i\leqslant m}\psi_i) \cup fn(\bigwedge_{1\leqslant j\leqslant n}\psi'_j) \cup fn(\phi)$. Assume that $\sigma \wedge \sigma' \leqslant \phi$, $\sigma \nleqslant \phi$ and $\sigma' \nleqslant \phi$. Then for all i $(1 \leqslant i \leqslant m)$ $\psi_i \nleqslant^{\mathsf{N}} \phi$ and for all j $(1 \leqslant j \leqslant n)$, $\psi'_j \nleqslant^{\mathsf{N}} \phi$. This, by Lemma 3.11(1), implies that for all i $(1 \leqslant i \leqslant m)$ $\overline{v\vec{c}}\Gamma_{\phi,w}\left[\psi_i\right] \simeq \omega$, and for all j, $1 \leqslant j \leqslant n$, $\overline{v\vec{c}}\Gamma_{\phi,w}\left[\psi'_j\right] \simeq \omega$. Therefore $\bigwedge_{1\leqslant i\leqslant m}\overline{v\vec{c}}\Gamma_{\phi,w}\left[\psi_i\right] \wedge \bigwedge_{1\leqslant j\leqslant n}\overline{v\vec{c}}\Gamma_{\phi,w}\left[\psi'_j\right] \simeq \omega$. This is a contradiction since $\sigma \wedge \sigma' \leqslant \phi$ implies $\bigwedge_{1\leqslant i\leqslant m}\overline{v\vec{c}}\Gamma_{\phi,w}\left[\psi_i\right] \wedge \bigwedge_{1\leqslant j\leqslant n}\overline{v\vec{c}}\Gamma_{\phi,w}\left[\psi'_j\right] \simeq \overline{v\vec{c}}\Gamma_{\phi,w}\left[\sigma \wedge \sigma'\right] \leqslant \overline{v\vec{c}}\Gamma_{\phi,w}\left[\phi\right]$, and $\overline{v\vec{c}}\Gamma_{\phi,w}\left[\phi\right] \not\simeq \omega$.
2 Let τ be a prime type. By Proposition 3.7, $\tau \simeq \psi_1 \wedge \cdots \wedge \psi_m$ where $\psi_1, \cdots, \psi_m$ are normal types. The primality of τ implies that $\psi_i \leqslant \tau$ for some i $(1 \leqslant i \leqslant m)$. Moreover, since $\tau \simeq \psi_1 \wedge \cdots \wedge \psi_m \leqslant \psi_j$ for all j, $1 \leqslant j \leqslant m$, we also have $\tau \leqslant \psi_i$. Therefore $\tau \simeq \psi_i$. □

Now we can prove the key relations between subtyping and normal subtyping.

Theorem 3.14. Let $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_m$ and $\tau \simeq \psi_1 \wedge \cdots \wedge \psi_n$, where $\phi_1, \cdots, \phi_m, \psi_1, \cdots, \psi_n$ are normal types. $\sigma \leqslant \tau$ if and only if for all i $(1 \leqslant i \leqslant n)$ there is j $(1 \leqslant j \leqslant m)$ such that $\phi_j \leqslant^{\mathsf{N}} \psi_i$.

Proof. The 'if' implication is obvious. For the 'Only If', let $\phi_1 \wedge \cdots \wedge \phi_m \leqslant \psi_1 \wedge \cdots \wedge \psi_n$. Then given any i $(1 \leqslant i \leqslant n)$, we have that $\phi_1 \wedge \cdots \wedge \phi_m \leqslant \psi_i$. By Theorem 3.13(1), for some j $(1 \leqslant j \leqslant m)$, $\phi_j \leqslant^{\mathsf{N}} \psi_i$. □

We say that types $\sigma_1, \ldots, \sigma_n$ are *incomparable* if for no $i \neq j$ $(1 \leqslant i, j \leqslant n)$, $\sigma_i \leqslant \sigma_j$.

Corollary 3.15. The decomposition of a type as an intersection of incomparable normal types is unique modulo $\simeq^{\mathsf{N}}$.

Proof. Let $\phi_1 \wedge \cdots \wedge \phi_m$ and $\psi_1 \wedge \cdots \wedge \psi_n$ be two intersections of incomparable, normal types. From Theorem 3.14, we have that $\phi_1 \wedge \cdots \wedge \phi_n \simeq \psi_1 \wedge \cdots \wedge \psi_m$ implies $m = n$, and for all i $(1 \leqslant i \leqslant m)$ there is j $(1 \leqslant j \leqslant m)$ such that $\phi_i \simeq^{\mathsf{N}} \psi_j$. □

4. The type assignment system

In this section we define the type assignment system characterizing the π-calculus, and we prove the property of subject expansion.

Definition 4.1. (The type assignment system)

$$\begin{array}{llll}
(\omega) & P : \omega & (\leqslant) & \dfrac{P : \sigma \quad \sigma \leqslant \tau}{P : \tau} \\[2ex]
(\langle\,\rangle\,\mathrm{I}) & \dfrac{P : \sigma}{x(y).P : \langle x(y)\rangle\sigma} & (\langle\bar{\ }\rangle\,\mathrm{I}) & \dfrac{P : \sigma}{\bar{x}y.P : \langle \bar{x}y\rangle\sigma} \\[2ex]
(\wr\,\mathrm{I}_{|}) & \dfrac{P : \sigma \quad Q : \tau}{P|Q : \sigma \wr \tau} & (\wr\,\mathrm{I}_{!}) & \dfrac{P : \sigma \quad !P : \tau}{!P : \sigma \wr \tau} \\[2ex]
(\nu\mathrm{I}) & \dfrac{P : \sigma}{\nu x P : \nu x\, \sigma} & ([\,]\mathrm{I}) & \dfrac{P : \sigma}{[x = y]P : [x = y]\sigma}
\end{array}$$

We write $\vdash P : \sigma$ if the statement $P : \sigma$ can be derived from the axioms and rules of Definition 4.1. We call deductions in which the rule $(\leqslant)$ is not used *$\leqslant$-free deductions*, and use $\vdash_{-\leqslant}$ to denote derivation in the system without the rule $(\leqslant)$.

As we can see, the system only has introduction rules for the various constructors. Elimination of prefixes and parallel composition is done using the inclusion rule $(\leqslant)$. Such a rule also acts as an $\wedge$ introduction and elimination (*cf.* Proposition 4.12).

Example 4.2. The two main features of π-calculus are the mobility (that is, the ability of passing channel names) and the creation of names. As an example of name creation, consider $M \equiv\, !\nu a\, \bar{b}a.\mathbf{0}$. The process M can generate an infinite number of new names. To M we can assign all the normal types

$$\langle \bar{b}(a_1)\rangle \cdots \langle \bar{b}(a_n)\rangle\omega$$

for any $n \geqslant 0$. Clearly these types express the property of giving as outputs on channel b a finite but unbounded number of local names.

The axioms **A11** and rules **R1** for type inclusion allow us to postpone the use of the rule $(\leqslant)$ till the end of deductions. Therefore every deduction can be transformed into a deduction in which the rule $(\leqslant)$ is used exactly once, and it is the last applied rule.

Proposition 4.3. ($\leqslant$-free deduction property)

1 $\vdash P : \sigma$ imply $\vdash_{-\leqslant} P : \tau$ for some $\tau \leqslant \sigma$.
2 $\vdash_{-\leqslant} P : \sigma$ implies $fn(\sigma) \subseteq fn(P)$.

Proof. Both points can be proved by induction on deductions.
For 1, if the last rule applied is $(\langle\,\rangle\,\mathrm{I})$, we have

$$(\langle\,\rangle\,\mathrm{I})\quad \frac{Q : \tau}{x(y).Q : \langle x(y)\rangle\tau}.$$

By the induction hypothesis, $\vdash_{-\leqslant} Q : \tau'$ for some $\tau' \leqslant \tau$. So by $(\langle\,\rangle\,\mathrm{I})$, we get $\vdash_{-\leqslant} x(y).Q : \langle x(y)\rangle\tau'$, where $\langle x(y)\rangle\tau' \leqslant \langle x(y)\rangle\tau$.

The other cases are similar.

The proof of 2 is immediate. □

If P is a process that does not contain !, it is easy to define its *principal type*, that is, a type $\wp(P)$ such that $\vdash P : \wp(P)$ and for all types σ, $\vdash P : \sigma$ implies $\wp(P) \leqslant \sigma$. Really, P admits a unique $\leqslant$-free deduction that derives its principal type.

Definition 4.4. (Principal type) Let P be a process not containing !. We define $\wp(P)$ by structural induction:

$$\begin{array}{lcl}
\wp(\mathbf{0}) & = & \omega\\
\wp(\alpha.M) & = & \langle\alpha\rangle\wp(M)\\
\wp(M|N) & = & \wp(M)\wr\wp(N)\\
\wp(\nu xM) & = & \nu x\wp(M)\\
\wp([x=y]M) & = & [x=y]\wp(M).
\end{array}$$

Proposition 4.5. (Principal type property) Let P be a process not containing !. Then

1 $\vdash_{-\leqslant} P : \wp(P)$, and
2 $\vdash P : \sigma$ implies $\wp(P) \leqslant \sigma$.

Proof.

1 The proof is by structural induction on P.
2 From Proposition 4.3 $\vdash_{-\leqslant} P : \tau$ for some $\tau \leqslant \sigma$. The proof is by induction on the deduction of $\vdash_{-\leqslant} P : \tau$. □

Remark 4.6. From the principal type property, and the decidability of $\leqslant$ (see Remark 3.2) we get the decidability of our type assignment system for processes without occurrences of !. Of course this does not hold for the whole set of processes.

The given type assignment system has the obvious structural properties, which can be shown by simple induction on derivations.

Lemma 4.7. (Structural properties of deductions)

1 $\vdash \mathbf{0} : \sigma$ if and only if $\omega \simeq \sigma$.
2 $\vdash x(y).P : \sigma$ if and only if for some τ, $\vdash_{-\leqslant} P : \tau$ and $\langle x(y)\rangle\tau \leqslant \sigma$.
3 $\vdash \bar{x}y.P : \sigma$ if and only if for some τ, $\vdash_{-\leqslant} P : \tau$ and $\langle \bar{x}y\rangle\tau \leqslant \sigma$.
4 $\vdash P|Q : \rho$ if and only if for some σ, τ, $\vdash_{-\leqslant} P : \sigma$, $\vdash_{-\leqslant} Q : \tau$ and $\sigma \wr \tau \leqslant \rho$.
5 $\vdash \nu x P : \sigma$ if and only if for some τ, $\vdash_{-\leqslant} P : \tau$ and $\nu x \tau \leqslant \sigma$.
6 $\vdash [x=y]P : \sigma$ if and only if for some τ, $\vdash_{-\leqslant} P : \tau$ and $[x=y]\tau \leqslant \sigma$.
7 $\vdash\, !P : \sigma$ if and only if for some $\sigma_1,\ldots,\sigma_n$ $(n \geqslant 1)$, $\vdash_{-\leqslant} P : \sigma_1$, …, $\vdash_{-\leqslant} P : \sigma_n$ and $\sigma_1 \wr \cdots \wr \sigma_n \leqslant \sigma$.

Proof. All cases follow easily from Proposition 4.3(1). The most interesting is Point 7. Let $\vdash\, !P : \sigma$. By Proposition 4.3(1), $\vdash_{-\leqslant}\, !P : \tau$ for some $\tau \leqslant \sigma$.

Let m be the number of applications of Rule ($\wr$ I$_!$) with $!P$ as subject of the conclusion that occur at the end of a $\leqslant$-free deduction of $!P : \tau$. We prove, by induction on m, that τ is of the form $\sigma_1 \wr \cdots \wr \sigma_m \wr \omega$ and $\vdash_{-\leqslant} P : \sigma_1, \ldots, \vdash_{-\leqslant} P : \sigma_m$.

The case $m = 0$ is trivial, in fact in this case we can just use Rule (ω).

If $m \geqslant 1$, then τ is of the form $\sigma_1 \wr \rho$ for some σ_1, ρ such that $\vdash_{-\leqslant} P : \sigma_1$, and there is a $\leqslant$-free deduction of $!P : \rho$ that has exactly $m-1$ applications of Rule ($\wr$ I$_!$) at the end. By the induction hypothesis, ρ is of the form $\sigma_2 \wr \cdots \wr \sigma_m \wr \omega$ and $\vdash_{-\leqslant} P : \sigma_2, \ldots, \vdash_{-\leqslant} P : \sigma_m$. □

Lemma 4.7 says that the types of a term can be obtained in a uniform way from the types of its subterms, and this will guarantee the compositionality of the filter model we will build in the next section.

Remark 4.8. As noted at the beginning of Section 3, the intersection operator corresponds to the internal choice operator $\oplus$ (see Remark 2.10). In fact, if

$$M \equiv \nu x\,(\bar{x}y.P|\bar{x}y.Q|x(z).\mathbf{0})$$

(where $x \notin fn(P) \cup fn(Q)$), it is straightforward to verify using Lemma 4.7 that M has type ρ if and only if $\rho \simeq \sigma \wedge \tau$, where σ is a type of P and τ is a type of Q. The crucial observation is that whenever $\sigma' \wedge \tau' \leqslant \rho$ we can find σ, τ such that $\sigma' \leqslant \sigma$, $\tau' \leqslant \tau$ and $\sigma \wedge \tau \simeq \rho$. This can be easily proved by taking the prime decomposition of σ, τ, ρ and by using Theorem 3.14.

We now state some properties of deductions that will be useful later.

Lemma 4.9.

1 $\vdash P : \sigma$ implies $\vdash P\{y/x\} : \sigma\{y/x\}$.
2 $\vdash_{-\leqslant} P\{y/x\} : \tau$ implies that τ is of the form $\sigma\{y/x\}$, for some σ such that $\vdash_{-\leqslant} P : \sigma$.
3 $\vdash P\{y/x\} : \sigma\{y/x\}$ implies $\vdash P : [x = y]\sigma$.

Proof. Points 1 and 2 can be shown by induction on deductions. For Point 3, we have $\vdash_{-\leqslant} P\{y/x\} : \tau$ with $\tau \leqslant \sigma\{y/x\}$ by Proposition 4.3(1). By Point 2 this implies that τ is of the form $\rho\{y/x\}$, for some ρ such that $\vdash_{-\leqslant} P : \rho$. Now $\rho \leqslant [x = y]\rho \simeq [x = y](\rho\{y/x\}) \leqslant [x = y](\sigma\{y/x\}) \simeq [x = y]\sigma$ by Axioms **A11**(3) and **A6**(4), and Rule **R1**(3), so we conclude $\vdash P : [x = y]\sigma$. □

Since we are in a ‘may’ perspective, it is natural that a process P offer all the communications offered by one of its reducts Q (may be more). At the type assignment level this means that types are preserved under subject expansion. Of course subject reduction should not hold, since reducing P with the rule COM may produce a process that offers fewer communications (and thus with fewer types).

Lemma 4.10. (Subject congruence) $\vdash P : \rho$ and $P \equiv Q$ imply $\vdash Q : \rho$.

Proof. By cases on the definition of $\equiv$ (Definition 2.2), using Lemma 4.7. The only interesting case is the third axiom of **A3**.

In one direction we have to show that $\vdash \nu x\,P|Q : \rho$ implies $\vdash \nu x\,(P|Q) : \rho$ under the condition $x \notin fn(Q)$. By Lemma 4.7, we have $\vdash_{-\leqslant} P : \sigma$, $\vdash_{-\leqslant} Q : \tau$ and $\nu x\,\sigma \wr \tau \leqslant \rho$. By ($\wr$ I$_|$) and (νI) we get $\vdash \nu x\,(P|Q) : \nu x\,(\sigma \wr \tau)$. Now, by Lemma 4.3(2), $x \notin fn(\tau)$, which implies (by Axiom **A3**(2)) $\nu x\,(\sigma \wr \tau) \simeq \nu x\,\sigma \wr \tau$. So by Rule ($\leqslant$), we conclude $\vdash \nu x\,(P|Q) : \rho$. Similarly, we can prove that $\vdash \nu x\,(P|Q) : \rho$ implies $\vdash \nu x\,P|Q : \rho$. □

Theorem 4.11. (Subject expansion) $\vdash P : \rho$ and $Q \longrightarrow\!\!\!\!\rightarrow P$ imply $\vdash Q : \rho$.

Proof. Remember that by definition $\longrightarrow\!\!\!\!\rightarrow \; = \overset{+}{\longrightarrow} \cup \equiv$, where $\overset{+}{\longrightarrow}$ is the transitive closure of $\longrightarrow$.

If $Q \equiv P$, the result follows by Lemma 4.10. If $Q \overset{+}{\longrightarrow} P$, the proof is by induction on the length of the expansion sequence. The only non-trivial case is Rule COM.

Let $x(y).M|\bar{x}z.N \longrightarrow M\{z/y\}|N$. Now, by Lemma 4.7(4), there are σ', τ such that $\vdash_{-\leqslant} M\{z/y\} : \sigma'$, $\vdash_{-\leqslant} N : \tau$ and $\sigma' \wr \tau \leqslant \rho$. By Lemma 4.9(2), σ' can be written as $\sigma\{z/y\}$, where σ is such that $\vdash_{-\leqslant} M : \sigma$. From $\vdash_{-\leqslant} M : \sigma$, by ($\langle\,\rangle$ I), we get $\vdash_{-\leqslant} x(y).M : \langle x(y)\rangle\sigma$, and from $\vdash_{-\leqslant} N : \tau$, by ($\langle\bar{\,}\rangle$ I), we get $\bar{x}z.N : \langle\bar{x}z\rangle\tau$. Then by ($\wr$ I$_|$), we deduce $\vdash_{-\leqslant} x(y).M|\bar{x}z.N : \langle x(y)\rangle\sigma \wr \langle\bar{x}z\rangle\tau$. Finally, by ($\leqslant$), observing that $\langle x(y)\rangle\sigma \wr \langle\bar{x}z\rangle\tau \leqslant \sigma\{z/y\} \wr \tau$ by Axioms **A8**(2) and **A11**(3), we conclude $\vdash_{-\leqslant} x(y).M|\bar{x}z.N : \rho$. □

It is interesting to notice that the standard rule of intersection introduction is admissible in our system.

Proposition 4.12. For the type assignment system of Definition 4.1, the rule

$$(\wedge\mathrm{I})\quad \frac{P : \sigma \quad P : \tau}{P : \sigma \wedge \tau}$$

is an admissible rule.

Proof. We prove that $\vdash P : \sigma$ and $\vdash P : \tau$ imply $\vdash P : \sigma \wedge \tau$ by induction on the structure of P.

Let P be $x(y).Q$. By Lemma 4.7(2) there are σ', τ' such that $\vdash Q : \sigma'$, $\vdash Q : \tau'$ and $\langle x(y)\rangle\sigma' \leqslant \sigma$, $\langle x(y)\rangle\tau' \leqslant \tau$. By induction, $\vdash Q : \sigma' \wedge \tau'$, which implies $\vdash x(y).Q : \langle x(y)\rangle(\sigma' \wedge \tau')$ by Rule ($\langle\,\rangle$ I). By Axiom **A10**(2) and Rule **R1**(5), we have $\langle x(y)\rangle(\sigma' \wedge \tau') \simeq \langle x(y)\rangle\sigma' \wedge \langle x(y)\rangle\tau' \leqslant \sigma \wedge \tau$.

Let P be $!Q$. By Lemma 4.7(7), there are $\sigma_1, \ldots, \sigma_m$ such that $\vdash Q : \sigma_i$ $(1 \leqslant i \leqslant m)$ and $\sigma_1 \wr \cdots \wr \sigma_m \leqslant \sigma$. Again by Lemma 4.7(7), there are $\tau_1, \ldots, \tau_n$ such that $\vdash Q : \tau_j$ $(1 \leqslant j \leqslant n)$ and $\tau_1 \wr \cdots \wr \tau_n \leqslant \tau$. By induction, $\vdash Q : \sigma_i \wedge \tau_j$ $(1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n)$, which imply $\vdash\, !Q : \wr_{1\leqslant i\leqslant m,\, 1\leqslant j\leqslant n}\, (\sigma_i \wedge \tau_j)$ by Rule ($\wr$ I$_!$). Using Axioms **A10**(3), **A11**(4) and **A2**(3), it is easy to check that $\wr_{1\leqslant i\leqslant m,\, 1\leqslant j\leqslant n}\, (\sigma_i \wedge \tau_j) \leqslant (\sigma_1 \wr \cdots \wr \sigma_m) \wedge (\tau_1 \wr \cdots \wr \tau_n)$. So we can conclude $\vdash\, !Q : \sigma \wedge \tau$ by ($\leqslant$).

All other cases are similar to the case of the input prefix. □

We end this section by introducing a subclass of processes that are characterized by bodies. Lemma 4.14 shows that this class contains interesting processes, In particular, all mono processes. Remember that, by definition, a process P is mono if and only if it contains at most one free name.

Definition 4.13. (Body processes) We say that a process P is a *body* process if $\vdash_{-\leqslant} P : \sigma$ implies $\sigma \simeq \beta_1 \wedge \cdots \wedge \beta_n$ $(n \geqslant 1)$, where $\beta_1, \cdots, \beta_n$ are bodies.

Lemma 4.14.

1 Every process that does not contain matchings, parallel compositions or replications is a body process.

2 Every mono process is a body process.

Proof.

1 Let P be a process that does not contain matchings, parallel compositions or replications. Assume that $\vdash_{-\leqslant} P : \sigma$, it is easy to show by induction on the deduction that σ is a body.

2 Let P be a mono process. Assume that $\vdash_{-\leqslant} P : \sigma$, then by Proposition 4.3(2), $fn(\sigma)$ contains at most one name. By Proposition 3.7, $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_m$ and $fn(\sigma) \supseteq fn(\phi_1 \wedge \cdots \wedge \phi_m)$, where $\phi_1, \cdots, \phi_m$ are normal types. But a normal type with at most one free name is a body and therefore each ϕ_i is a body. □

5. The filter model

In this section we first introduce the filter model for the π-calculus defined in Section 2 and then we prove its adequacy and full abstraction with respect to the operational semantics.

Let $\langle D; \sqsubseteq \rangle$ be a preorder. A subset L of D is a *filter* if L is a non-empty upper set, that is, $l \in L$ and $l \sqsubseteq l'$ imply $l' \in L$, and every finite subset of L has a lower bound in L. Consider the set $\mathscr{T}$ of types with the inclusion $\leqslant$ defined in Section 3. The lower bound of a finite non-empty set of types is the intersection of the types in the set.

We can observe that the set $\mathscr{F}(\mathscr{T})$ of filters over $\mathscr{T}$ is a model of Π in the following sense. For all M define

$$[\![M]\!] = \{\sigma \in \mathscr{T} \mid \vdash M : \sigma\}.$$

From Rules (ω), ($\leqslant$) and the admissibility of ($\wedge$I), we have that $[\![M]\!] \in \mathscr{F}(\mathscr{T})$ for all M. Subject expansion can now be rephrased into the following statement:

$$\text{if } M \longrightarrow\!\!\!\!\!\rightarrow N \text{ then } [\![M]\!] \supseteq [\![N]\!].$$

The filter model is naturally ordered by subset inclusion. The inclusion on filters induces an ordering on terms.

Definition 5.1. Let $M, N \in \Pi$. $M \sqsubseteq_F N$ if and only if $[\![M]\!] \subseteq [\![N]\!]$.

The order relation $\sqsubseteq_F$ can be easily characterized by means of the deducibility of types as follows.

Proposition 5.2. Let $M, N \in \Pi$. $M \sqsubseteq_F N$ if and only if, for all σ, $\vdash M : \sigma$ implies $\vdash N : \sigma$.

We will prove that the filter model exactly mirrors the operational semantics, that is, that it is adequate and fully abstract.

The adequacy proof requires a double induction on types and deductions. Following a standard methodology, we split this induction by introducing a realizability interpretation of types as sets of terms. The underlying idea is that a term M belongs to the interpretation of a type σ if and only if σ can be derived for M.

First we give an interpretation of normal types, and then we build the interpretation of all types, taking into account Theorem 3.14 and Corollary 3.15. The interpretation of normal types is given by structural induction.

Definition 5.3. If ϕ is a normal type, the realizability of ϕ is defined by:

$$\begin{aligned}
[\![\omega]\!] &= \Pi \\
[\![\langle x(y)\rangle\beta]\!] &= \{M \in \Pi \mid M \twoheadrightarrow \nu\vec{c}(x(y).P|Q) \text{ for some } P, Q, \vec{c} \text{ such that} \\
&\quad\ \nu\vec{c}(P|Q) \in [\![\beta]\!] \text{ and } x \notin \vec{c}\,\} \\
[\![\langle \bar{x}y\rangle\beta]\!] &= \{M \in \Pi \mid M \twoheadrightarrow \nu\vec{c}(\bar{x}y.P|Q) \text{ for some } P, Q, \vec{c} \text{ such that} \\
&\quad\ \nu\vec{c}(P|Q) \in [\![\beta]\!] \text{ and } x, y \notin \vec{c}\,\} \\
[\![\langle x[z = (y)]\rangle\beta]\!] &= \{M \in \Pi \mid M \twoheadrightarrow \nu\vec{c}(x(y).P|Q) \text{ for some } P, Q, \vec{c} \text{ such that} \\
&\quad\ \nu\vec{c}(P|Q)\{z/y\} \in [\![\beta]\!] \text{ and } x, z \notin \vec{c}\,\} \\
[\![\langle \bar{x}(y)\rangle\beta]\!] &= \textstyle\bigcup_{z\in\mathcal{N}} [\![\langle \bar{x}z\rangle(\beta\{z/y\})]\!] \cup \\
&\quad\ \{M \in \Pi \mid M \twoheadrightarrow \nu y \nu\vec{c}(\bar{x}y.P|Q) \text{ for some } P, Q, \vec{c} \text{ such that} \\
&\quad\ \nu\vec{c}(P|Q) \in [\![\beta]\!] \text{ and } x \notin \vec{c}\,\} \\
[\![[x = y]\psi]\!] &= \{M \in \Pi \mid M\{x/y\} \in [\![\psi]\!]\}.
\end{aligned}$$

To define the realizability of an arbitrary type, we need the soundness of the normal type inclusion relation with respect to the interpretation of normal types (Theorem 5.5). To prove this, we relate the interpretation of ϕ to that of $\phi\{x/y\}^{\mathsf{N}}$.

Lemma 5.4.

1 If $[x = y]\phi$ is normal, then $[\![[x = y]\phi]\!] = [\![[y = x](\phi\{y/x\})]\!]$.
2 If $[x = y][a = b]\phi$ is normal, then $[\![[x = y][a = b]\phi]\!] = [\![[a = b][x = y]\phi]\!]$.
3 If ϕ is normal, then $[\![\phi]\!] = [\![\mathsf{rep}(x, \phi)]\!]$.
4 If β is a body, then $[\![\beta\{x/y\}]\!] = \{M\{x/y\} \mid M \in [\![\beta]\!]\}$.
5 If ϕ is normal, then $[\![\phi\{x/y\}^{\mathsf{N}}]\!] = \{M\{x/y\} \mid M \in [\![\phi]\!]\}$.

Proof.

1 Observe first that $[y = x](\phi\{y/x\})$ is normal. It is easy to prove by induction on ϕ that $M\{x/y\} \in [\![\phi]\!]$ if and only if $M\{y/x\} \in [\![\phi\{y/x\}]\!]$. Then $M \in [\![[x = y]\phi]\!] \iff$ (by definition) $M\{x/y\} \in [\![\phi]\!] \iff M\{y/x\} \in [\![\phi\{y/x\}]\!] \iff$ (again by definition) $M \in [\![[y = x](\phi\{y/x\})]\!]$.
2 Notice that by definition of normal type $y \neq a, b$ and $x \neq b$, therefore $M\{x/y\}\{a/b\} \equiv M\{a/b\}\{x/y\}$. So we conclude the result using the definition of realizability of matching (last case).
3 Let $\phi = \overrightarrow{[a = b]}\beta$. If $\mathsf{rep}(x, \phi) = \phi$ the result is obvious. Otherwise $[c = x] \in \overrightarrow{[a = b]}$. Let $\psi = (\overrightarrow{[a = b]} - [c = x])\beta$. By definition $\mathsf{rep}(x, \phi) = [x = c]\psi\{x/c\}$. Therefore by Point 2, $[\![\phi]\!] = [\![[c = x]\psi]\!]$ and by Point 1, $[\![[c = x]\psi]\!] = [\![[x = c]\psi\{x/c\}]\!]$.
4 Easy by structural induction on β.
5 Let $\psi = \mathsf{rep}(x, \mathsf{rep}(y, \phi))$. By Point 3 we have $[\![\phi]\!] = [\![\psi]\!]$. So it is sufficient to prove, by structural induction on ψ and using the inductive definition of $\{\cdot/\cdot\}^{\mathsf{R}}$ (Definition 3.5), that: $M \in [\![\psi]\!] \iff M\{x/y\} \in [\![\psi\{x/y\}^{\mathsf{R}}]\!]$.
   — If ψ is a body, then the result follows by Point 4.
   — If $\psi = [x = y]\psi'$, then $\psi\{x/y\}^{\mathsf{R}} = \psi'$. Moreover,
   $M \in [\![\psi]\!] \iff$ (by definition) $M\{x/y\} \in [\![\psi']\!] = [\![\psi\{x/y\}^{\mathsf{R}}]\!]$.
   — The other cases follow immediately from the induction hypothesis. □

Theorem 5.5. Let ϕ and ψ be normal types. $\phi \leqslant^{\mathsf{N}} \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.

Proof. The proof is by induction on the length of the derivation $\phi \leqslant^{\mathsf{N}} \psi$. We only consider the interesting cases.

— $\langle x(y)\rangle\beta \leqslant^{\mathsf{N}} \langle x[z=(y)]\rangle\beta\{z/y\}$ where $z \neq y$.
If $M \in [\![\langle x(y)\rangle\beta]\!]$ then $M \longrightarrow\!\!\!\!\!\rightarrow \overrightarrow{\nu c}(x(y).P|Q)$ for some $P, Q, \vec{c}$ such that $\overrightarrow{\nu c}(P|Q) \in [\![\beta]\!]$ and $x \notin \vec{c}$. We can assume that $z \notin \vec{c}$.
$\overrightarrow{\nu c}(P|Q) \in [\![\beta]\!]$ implies $\overrightarrow{\nu c}(P|Q)\{z/y\} \in [\![\beta\{z/y\}]\!]$ by Lemma 5.4(4). By definition, we conclude $M \in [\![\langle x[z=(y)]\rangle\beta\{z/y\}]\!]$.

— $\phi \leqslant^{\mathsf{N}} [x=y](\phi\{x/y\}^{\mathsf{N}})$. If $M \in [\![\phi]\!]$, then $M\{x/y\} \in [\![\phi\{x/y\}^{\mathsf{N}}]\!]$ by Lemma 5.4(5). Then, by definition, $M \in [\![[x=y]\phi\{x/y\}^{\mathsf{N}}]\!]$. □

Now we can define the interpretation of all types.

Definition 5.6. If σ is a type, the realizability of σ is

$$[\![\sigma]\!] = \bigcap_{1\leqslant i\leqslant n} [\![\phi_i]\!],$$

where ϕ_i $(1 \leqslant i \leqslant n)$ are incomparable normal types such that $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_n$.

Notice that, as shown in Corollary 3.15, the decomposition of a type as an intersection of incomparable normal types is unique modulo $\simeq^{\mathsf{N}}$. This, together with Theorem 5.5, ensures that $[\![\sigma]\!]$ is well defined, in the sense that it does not depend on the choice of the incomparable normal types ϕ_i $(1 \leqslant i \leqslant n)$ such that $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_n$. Moreover, from Theorem 5.5, we derive that we do not need a decomposition in incomparable types. This is because, for any ϕ'_i $(1 \leqslant i \leqslant m)$ normal and ϕ_i $(1 \leqslant i \leqslant n)$ normal, such that $\sigma \simeq \phi'_1 \wedge \cdots \wedge \phi'_m$ and $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_n$, we have that

$$\bigcap_{1\leqslant i\leqslant n} [\![\phi_i]\!] = \bigcap_{1\leqslant i\leqslant m} [\![\phi'_i]\!].$$

Theorem 5.7. Let σ and τ be any types. Then $\sigma \leqslant \tau$ implies $[\![\sigma]\!] \subseteq [\![\tau]\!]$.

Proof. The proof is easy from Definition 5.6 and Theorems 3.14 and 5.5. □

It is easy to prove that realizability sets are closed under subject expansion and under parallel composition.

Lemma 5.8.

1 If $N \in [\![\sigma]\!]$ and $M \longrightarrow\!\!\!\!\!\rightarrow N$, then $M \in [\![\sigma]\!]$.
2 $M \in [\![\sigma]\!]$ implies that $M|N \in [\![\sigma]\!]$ for all N.

Proof. By Definition 5.6, it is sufficient to consider the case of normal types. Both proofs are by structural induction on normal types.

1 All cases follow from the transitivity of $\longrightarrow\!\!\!\!\!\rightarrow$ and from induction. For the last case, notice that if $M \longrightarrow\!\!\!\!\!\rightarrow N$, then $M\{x/y\} \longrightarrow\!\!\!\!\!\rightarrow N\{x/y\}$.
2 If $\phi = \langle x(y)\rangle\beta$, then $M \longrightarrow\!\!\!\!\!\rightarrow \overrightarrow{\nu c}(x(y).P|Q)$ for some $P, Q, \vec{c}$ such that $\overrightarrow{\nu c}(P|Q) \in [\![\beta]\!]$ and $x \notin \vec{c}$. By induction, $\overrightarrow{\nu c}(P|Q)|N \in [\![\beta]\!]$. We can assume that $\vec{c} \notin fn(N)$. So $\overrightarrow{\nu c}(P|Q|N) \in [\![\beta]\!]$ by Point 1, since $\overrightarrow{\nu c}(P|Q)|N \equiv \overrightarrow{\nu c}(P|Q|N)$. Now we have that $M|N \longrightarrow\!\!\!\!\!\rightarrow \overrightarrow{\nu c}(x(y).P|Q|N)$ implies $M|N \in [\![\phi]\!]$.
If $\phi = \langle x[z=(y)]\rangle\beta$, then we have $M \longrightarrow\!\!\!\!\!\rightarrow \overrightarrow{\nu c}(x(y).P|Q)$ for some $P, Q, \vec{c}$ such that $\overrightarrow{\nu c}(P|Q)\{z/y\} \in [\![\beta]\!]$ and $x, z \notin \vec{c}$. By induction, $\overrightarrow{\nu c}(P|Q)\{z/y\}|N\{z/y\} \in$

$[\![\beta]\!]$. We can assume that $\vec{c} \notin fn(N)$. So $\overline{\nu\vec{c}}(P|Q|N)\{z/y\} \in [\![\beta]\!]$ by Point 1, since $\overline{\nu\vec{c}}(P|Q)\{z/y\}|N\{z/y\} \equiv \overline{\nu\vec{c}}(P|Q|N)\{z/y\}$. Now $M|N \longrightarrow\!\!\!\!\!\rightarrow \overline{\nu\vec{c}}(x(y).P|Q|N)$ implies $M|N \in [\![\phi]\!]$.
If $\phi = [x = y]\psi$, then $M\{x/y\} \in [\![\psi]\!]$. Then by the induction hypothesis, we have that $M\{x/y\}|N\{x/y\} \in [\![\psi]\!]$. So, since $M\{x/y\}|N\{x/y\} \equiv (M|N)\{x/y\}$, also $(M|N)\{x/y\} \in [\![\psi]\!]$ by Point 1. Therefore $M|N \in [\![[x = y]\psi]\!]$.
The proofs of the other cases are similar to the proof of the first one. □

According to Definition 5.3, some reduction properties characterize the processes that belong to the interpretations of normal types. We can extend this result to all types only partially, since we are able to give sufficient but not necessary conditions for the membership of processes to the interpretation of a (non-normal) type.

Lemma 5.9.

1 $M \longrightarrow\!\!\!\!\!\rightarrow \overline{\nu\vec{c}}(x(y).P|Q)$ for some $P, Q, \vec{c}$ such that $\overline{\nu\vec{c}}(P|Q) \in [\![\sigma]\!]$ and $x \notin \vec{c}$ imply $M \in [\![\langle x(y)\rangle\sigma]\!]$.
2 $M \longrightarrow\!\!\!\!\!\rightarrow \overline{\nu\vec{c}}(\bar{x}y.P|Q)$ for some $P, Q, \vec{c}$ such that $\overline{\nu\vec{c}}(P|Q) \in [\![\sigma]\!]$ and $x, y \notin \vec{c}$ imply $M \in [\![\langle \bar{x}y\rangle\sigma]\!]$.
3 $M \longrightarrow\!\!\!\!\!\rightarrow \nu x N$ for some N such that $N \in [\![\sigma]\!]$ implies $M \in [\![\nu x\sigma]\!]$.

Proof. By Theorem 5.7, it is sufficient to prove all points for normal types, observing that if $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_n$, then $\langle\alpha\rangle\sigma \simeq \langle\alpha\rangle\phi_1 \wedge \cdots \wedge \langle\alpha\rangle\phi_n$, and $\nu x\sigma \simeq \nu x\phi_1 \wedge \cdots \wedge \nu x\phi_n$, by **R1**(1) and **R1**(2).

1 Let $\phi = \overrightarrow{[a = b]}\beta$, where β is a body. If $\overline{\nu\vec{c}}(P|Q) \in [\![\phi]\!]$, then $\overline{\nu\vec{c}}(P|Q)\{\vec{a}/\vec{b}\} \in [\![\beta]\!]$ by Definition 5.3. Moreover, since $\overline{\nu\vec{c}}(P|Q) \in [\![\omega]\!]$,

$$M \in [\![\langle x(y)\rangle\omega]\!]. \qquad (*)$$

Now we have two different cases according to whether $y \notin \overrightarrow{[a = b]}$ or $y \in \overrightarrow{[a = b]}$.

(i) If $y \notin \overrightarrow{[a = b]}$, then $\langle x(y)\rangle\phi \simeq \langle x(y)\rangle\omega \wedge \overrightarrow{[a = b]}\langle x\{\vec{a}/\vec{b}\}(y)\rangle\beta$ by Lemma 3.6(3).
By (∗) and Theorem 5.7, it is sufficient to prove $M \in [\![\overrightarrow{[a = b]}\langle x\{\vec{a}/\vec{b}\}(y)\rangle\beta]\!]$.
Now $\overline{\nu\vec{c}}(P|Q)\{\vec{a}/\vec{b}\} \in [\![\beta]\!]$ implies $\overline{\nu\vec{c}}(x(y).P|Q)\{\vec{a}/\vec{b}\} \in [\![\langle x(y)\rangle\beta]\!]$ by definition, so we have that $\overline{\nu\vec{c}}(x(y).P|Q) \in [\![\overrightarrow{[a = b]}\langle x\{\vec{a}/\vec{b}\}(y)\rangle\beta]\!]$ again by definition. Then we conclude $M \in [\![\overrightarrow{[a = b]}\langle x\{\vec{a}/\vec{b}\}(y)\rangle\beta]\!]$ by Lemma 5.8(1).

(ii) If $[z = y]$ or $[y = z] \in \overrightarrow{[a = b]}$, let $\overrightarrow{[d = e]} = (\overrightarrow{[a = b]} - [z = y] - [y = z])\{z/y\}$. Then
$\langle x(y)\rangle\phi \simeq \langle x(y)\rangle\omega \wedge \overrightarrow{[d = e]}\langle x\{\vec{d}/\vec{e}\}[z = (y)]\rangle(\beta\{z/y\})$ by Lemma 3.6(3).
By (∗) and Theorem 5.7 it is sufficient for us to prove $M \in [\![\overrightarrow{[d = e]}\langle x\{\vec{d}/\vec{e}\}[z = (y)]\rangle\beta\{z/y\}]\!]$.
First observe that $\overline{\nu\vec{c}}(P|Q)\{\vec{d}/\vec{e}\}\{z/y\} \in [\![\beta\{z/y\}]\!]$ implies $\overline{\nu\vec{c}}(x(y).P|Q)\{\vec{d}/\vec{e}\} \in [\![\langle x[z = (y)]\rangle\{\vec{d}/\vec{e}\}\beta\{z/y\}]\!]$ by definition. So, again by definition, we have that $\overline{\nu\vec{c}}(x(y).P|Q) \in [\![\overrightarrow{[d = e]}\langle x\{\vec{d}/\vec{e}\}[z = (y)]\rangle\beta\{z/y\}]\!]$. Then we can conclude that $M \in [\![\overrightarrow{[d = e]}\langle x\{\vec{d}/\vec{e}\}[z = (y)]\rangle\beta\{z/y\}]\!]$ by Lemma 5.8(1).

2 The proof is as for the first case of Point 1 using Lemma 3.6(4).

3 The more interesting case is $\phi = \langle \bar{y}x\rangle\beta$. In this case $\nu^{\mathsf{N}}x\,\langle \bar{y}x\rangle\beta = \langle \bar{y}(x)\rangle\beta$ and $x \neq y$. If $N \in [\![\langle \bar{y}x\rangle\beta]\!]$, then, by definition, $N \longrightarrow\!\!\!\!\rightarrow \nu\vec{c}(\langle \bar{y}x\rangle.P|Q)$ for some $\vec{c}$, P, Q such that $\nu\vec{c}(P|Q) \in [\![\beta]\!]$ and $x, y \notin \vec{c}$. This implies $M \longrightarrow\!\!\!\!\rightarrow \nu x\nu\vec{c}(\langle \bar{y}x\rangle.P|Q)$. Therefore by definition, $M \in [\![\nu x.\phi]\!]$. □

Lemma 5.10.

1 $[\![\sigma\{x/y\}]\!] = \{M\{x/y\} \mid M \in [\![\sigma]\!]\}$.
2 $M\{x/y\} \in [\![\sigma]\!]$ implies $M \in [\![[x = y]\sigma]\!]$.
3 $M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\} \in [\![\sigma]\!]$ implies $M|N \in [\![\overrightarrow{[a = b]}\overrightarrow{[e = f]}\sigma]\!]$.

Proof. By Theorem 5.7, it is sufficient to prove the first two points for normal types, observing that if $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_n$, then $\sigma\{x/y\} \simeq \phi_1\{x/y\} \wedge \cdots \wedge \phi_n\{x/y\}$, and $[x = y]\sigma \simeq [x = y]\phi_1 \wedge \cdots \wedge [x = y]\phi_n$, by **R1**(3).

1 The proof is immediate from Lemmas 5.4(5), 3.6(2) and Theorem 5.7.
2 $M\{x/y\} \in [\![\phi]\!] \implies$ (by Lemma 5.4(5)) $M\{x/y\} \equiv (M\{x/y\})\{x/y\} \in [\![\phi\{x/y\}^{\mathsf{N}}]\!] \implies$ (by Definition 5.3, Axiom **A6**(4), Lemma 3.6(2), and Theorem 5.7) $M \in [\![[x = y](\phi\{x/y\}^{\mathsf{N}})]\!] = [\![[x = y]\phi]\!]$.
3 Let $\overrightarrow{[u = v]}$ be a normal[†] sequence such that $\mathscr{E}(\overrightarrow{[u = v]}) = \mathscr{E}(\overrightarrow{[a = b]}\overrightarrow{[e = f]})$ (Proposition 3.4(2) assures the existence of $\overrightarrow{[u = v]}$).
By Point 1, $M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\} \in [\![\sigma]\!]$ implies $(M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\})\{\vec{u}/\vec{v}\} \in [\![\sigma\{\vec{u}/\vec{v}\}]\!]$. Since $(M|N)\{\vec{u}/\vec{v}\} \equiv (M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\})\{\vec{u}/\vec{v}\}$, we have $(M|N)\{\vec{u}/\vec{v}\} \in [\![\sigma\{\vec{u}/\vec{v}\}]\!]$. So we obtain $M|N \in [\![\overrightarrow{[u = v]}\sigma]\!]$ by Point 2. We conclude $M|N \in [\![\overrightarrow{[a = b]}\overrightarrow{[e = f]}\sigma]\!]$ by Proposition 3.4(1) and Theorem 5.7. □

The next lemma states the key properties of the realizability interpretation needed to prove the soundness of realizability.

Lemma 5.11. Let $M \in [\![\sigma]\!]$:

1 $x(y).M \in [\![\langle x(y)\rangle\sigma]\!]$
2 $\bar{x}y.M \in [\![\langle \bar{x}y\rangle\sigma]\!]$
3 $\nu x\,M \in [\![\nu x\,\sigma]\!]$
4 $[x = y]M \in [\![[x = y]\sigma]\!]$
5 $N \in [\![\tau]\!]$ implies $M|N \in [\![\sigma \wr \tau]\!]$.

Proof. Points 1, 2, and 3 follow immediately from the corresponding Points of Lemma 5.9.

4 $M \in [\![\sigma]\!] \implies$ (by Lemma 5.10(1)) $M\{x/y\} \in [\![\sigma\{x/y\}]\!] \implies$ (since $([x = y]M)\{x/y\} \equiv M\{x/y\}$) $([x = y]M)\{x/y\} \in [\![\sigma\{x/y\}]\!] \implies$ (by Lemma 5.10(2)) $[x = y]M \in [\![[x = y]\sigma\{x/y\}]\!] \implies$ (by Axiom **A6**(4) and Theorem 5.7) $[x = y]M \in [\![[x = y]\sigma]\!]$.
5 Let $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_m$, $\tau \simeq \psi_1 \wedge \cdots \wedge \psi_n$ where ϕ_i $(1 \leqslant i \leqslant m)$, ψ_j $(1 \leqslant j \leqslant n)$ are normal. Then $\sigma \wr \tau \simeq \bigwedge_{1\leqslant i\leqslant m, 1\leqslant j\leqslant n}(\phi_i \wr \psi_j)$. Therefore $[\![\sigma \wr \tau]\!] = \bigcap_{1\leqslant i\leqslant m, 1\leqslant j\leqslant n}[\![\phi_i \wr \psi_j]\!]$ by Theorem 5.7. So we will prove that $M \in [\![\phi]\!]$ and $N \in [\![\psi]\!]$ imply $M|N \in [\![\phi \wr \psi]\!]$ when ϕ and ψ are normal. This proof is by structural induction on ϕ and ψ. We consider only the more interesting cases.

[†] We require $\overrightarrow{[u = v]}$ to be normal, since this assures us that the substitution $\{\vec{u}/\vec{v}\}$ is well defined.

(a) Let $\phi = \overrightarrow{[a=b]}\langle y[g=(x)]\rangle\beta$ and $\psi = \overrightarrow{[e=f]}\langle \bar{t}z\rangle\gamma$.
Then by Axiom **A8**(2)

$$\begin{aligned}\phi \wr \psi \quad \simeq \quad & (\overrightarrow{[a=b]}\langle y(x)\rangle([g=x]\beta \wr \psi)) \wedge (\overrightarrow{[e=f]}\langle \bar{t}z\rangle(\phi \wr \gamma))\wedge \\ & (\overrightarrow{[a=b]}\,\overrightarrow{[e=f]}[y=t]([g=z]\beta\{g/z\} \wr \gamma\{y/t\})).\end{aligned}$$

So we have to prove

— $M|N \in \llbracket\overrightarrow{[a=b]}\langle y(x)\rangle([g=x]\beta \wr \psi)\rrbracket$,

— $M|N \in \llbracket\overrightarrow{[e=f]}\langle \bar{t}z\rangle(\phi \wr \gamma)\rrbracket$, and

— $M|N \in \llbracket\overrightarrow{[a=b]}\,\overrightarrow{[e=f]}[y=t]([g=z]\beta\{g/x\} \wr \gamma\{y/t\})\rrbracket$.

If $M \in \llbracket\phi\rrbracket$, then $M\{\vec{a}/\vec{b}\} \in \llbracket\langle y[g=(x)]\rangle\beta\rrbracket$, that is, $M\{\vec{a}/\vec{b}\}\twoheadrightarrow\overrightarrow{\nu c}(y(x).P|Q)$, and $\overrightarrow{\nu c}(P|Q)\{g/x\} \in \llbracket\beta\rrbracket$ for some $P, Q, \vec{c}$, by Definition 5.3. Moreover, $N \in \llbracket\psi\rrbracket$ implies $N\{\vec{e}/\vec{f}\} \in \llbracket\langle\bar{t}z\rangle\gamma\rrbracket$, that is, $N\{\vec{e}/\vec{f}\}\twoheadrightarrow\overrightarrow{\nu d}(\bar{t}z.R|S)$ and $\overrightarrow{\nu d}(R|S) \in \llbracket\gamma\rrbracket$ for some $R, S, \vec{d}$, by the same definition.

Proof of $M|N \in \llbracket\overrightarrow{[a=b]}\langle y(x)\rangle([g=x]\beta \wr \psi)\rrbracket$.
From $\overrightarrow{\nu c}(P|Q)\{g/x\} \in \llbracket\beta\rrbracket$ we get, by definition, $\overrightarrow{\nu c}(P|Q) \in \llbracket[g=x]\beta\rrbracket$. Now $\overrightarrow{\nu c}(P|Q) \in \llbracket[g=x]\beta\rrbracket$ and $N \in \llbracket\psi\rrbracket$ imply $\overrightarrow{\nu c}(P|Q)|N \in \llbracket[g=x]\beta \wr \psi\rrbracket$ by induction. We can assume $\vec{c} \notin fn(N)$, so we get $\overrightarrow{\nu c}(P|Q)|N \equiv \overrightarrow{\nu c}(P|Q|N)$ and $\overrightarrow{\nu c}(P|Q|N) \in \llbracket[g=x]\beta \wr \psi\rrbracket$ by Lemma 5.8(1). By Lemma 5.9(1), this implies $M\{\vec{a}/\vec{b}\}|N \in \llbracket\langle y(x)\rangle([g=x]\beta \wr \psi)\rrbracket$, since $M\{\vec{a}/\vec{b}\}|N\twoheadrightarrow\overrightarrow{\nu c}(y(x).P|Q|N)$. We conclude, by Lemma 5.10(3), that $M|N \in \llbracket\overrightarrow{[a=b]}\langle y(x)\rangle([g=x]\beta \wr \psi)\rrbracket$.
The proof of $M|N \in \llbracket\overrightarrow{[e=f]}\langle\bar{t}z\rangle(\phi \wr \gamma)\rrbracket$ is similar.

Proof of $M|N \in \llbracket\overrightarrow{[a=b]}\,\overrightarrow{[e=f]}[y=t]([g=z]\beta\{g/x\} \wr \gamma\{y/t\})\rrbracket$.

$$\begin{aligned}\overrightarrow{\nu c}(P|Q)\{g/x\} \in \llbracket\beta\rrbracket \quad & \Longrightarrow \quad \overrightarrow{\nu c}(P|Q)\{g/x\}\{g/z\} \in \llbracket\beta\{g/z\}\rrbracket \text{ by Lemma 5.4(4)} \\ & \Longrightarrow \quad \overrightarrow{\nu c}(P|Q)\{z/x\}\{g/z\} \in \llbracket\beta\{g/z\}\rrbracket \\ & \Longrightarrow \quad \overrightarrow{\nu c}(P|Q)\{z/x\} \in \llbracket[g=z]\beta\{g/z\}\rrbracket \text{ by Definition 5.3} \quad (\circ) \\ \overrightarrow{\nu d}(R|S) \in \llbracket\gamma\rrbracket \quad & \Longrightarrow \quad \overrightarrow{\nu d}(R|S)\{y/t\} \in \llbracket\gamma\{y/t\}\rrbracket \text{ by Lemma 5.4(4)} \quad (\circ\circ)\end{aligned}$$

$$\begin{aligned}(\circ) \text{ and } (\circ\circ) \quad & \Longrightarrow \quad \overrightarrow{\nu c}(P|Q)\{z/x\}|\overrightarrow{\nu d}(R|S)\{y/t\} \in \llbracket[g=z]\beta\{g/z\} \wr \gamma\{y/t\}\rrbracket \\ & \qquad \text{by induction} \\ & \Longrightarrow \quad M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\}\{y/t\} \in \llbracket[g=z]\beta\{g/z\} \wr \gamma\{y/t\}\rrbracket \\ & \qquad \text{by Lemma 5.8(1) since} \\ & \qquad M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\}\{y/t\}\twoheadrightarrow\overrightarrow{\nu c}(P|Q)\{z/x\}|\overrightarrow{\nu d}(R|S)\{y/t\} \\ & \Longrightarrow \quad M|N \in \llbracket\overrightarrow{[a=b]}\,\overrightarrow{[e=f]}[y=t]([g=z]\beta\{g/x\} \wr \gamma\{y/t\})\rrbracket \\ & \qquad \text{by Lemma 5.10(3).}\end{aligned}$$

(b) Let $\phi = \overrightarrow{[a=b]}\langle y(x)\rangle\beta$ and $\psi = \overrightarrow{[e=f]}\langle\bar{t}(z)\rangle\gamma$.
Then by Axiom **A8**(2)

$$\begin{aligned}\phi \wr \psi \quad \simeq \quad & (\overrightarrow{[a=b]}\langle y(x)\rangle(\beta \wr \psi)) \wedge (\overrightarrow{[e=f]}\langle\bar{t}(z)\rangle(\phi \wr \gamma))\wedge \\ & (\overrightarrow{[a=b]}\,\overrightarrow{[e=f]}[y=t]\nu z\,(\beta\{z/x\} \wr \gamma\{y/t\})).\end{aligned}$$

So we have to prove

— $M|N \in [\![\overrightarrow{[a=b]}\langle y(x)\rangle(\beta \wr \psi)]\!]$,

— $M|N \in [\![\overrightarrow{[e=f]}\langle \bar{t}(z)\rangle(\phi \wr \gamma)]\!]$, and

— $M|N \in [\![\overrightarrow{[a=b]}\,\overrightarrow{[e=f]}[y=t]\nu z(\beta\{z/x\} \wr \gamma\{y/t\})]\!]$.

If $M \in [\![\phi]\!]$, then $M\{\vec{a}/\vec{b}\} \in [\![\langle y(x)\rangle\beta]\!]$, that is, $M\{\vec{a}/\vec{b}\} \longrightarrow\!\!\!\!\rightarrow \overrightarrow{\nu c}(y(x).P|Q)$ and $\overrightarrow{\nu c}(P|Q) \in [\![\beta]\!]$ for some $P, Q, \vec{c}$, by Definition 5.3. Moreover, $N \in [\![\psi]\!]$ implies $N\{\vec{e}/\vec{f}\} \in [\![\langle \bar{t}(z)\rangle\gamma]\!]$. Again, by Definition 5.3, this implies that: either

(i) $N\{\vec{e}/\vec{f}\} \in [\![\langle \bar{t}g\rangle\gamma\{g/z\}]\!]$ for some g, in other words, $N\{\vec{e}/\vec{f}\} \longrightarrow\!\!\!\!\rightarrow \overrightarrow{\nu d}(\bar{t}g.R|S)$ and $\overrightarrow{\nu d}(R|S) \in [\![\gamma\{g/z\}]\!]$ for some $R, S, \vec{d}$, or

(ii) $N\{\vec{e}/\vec{f}\} \longrightarrow\!\!\!\!\rightarrow \nu z\overrightarrow{\nu d}(\bar{t}z.R|S)$ and $\overrightarrow{\nu d}(R|S) \in [\![\gamma]\!]$ for some $R, S, \vec{d}$.

In both cases, (i) and (ii), the proof of $M|N \in [\![\overrightarrow{[a=b]}\langle y(x)\rangle(\beta \wr \psi)]\!]$ is similar to the proof of case (a).

Proof of $M|N \in [\![\overrightarrow{[e=f]}\langle \bar{t}(z)\rangle(\phi \wr \gamma)]\!]$.

Case (i). In this case $N\{\vec{e}/\vec{f}\} \longrightarrow\!\!\!\!\rightarrow \overrightarrow{\nu d}(\bar{t}g.R|S)$ and $\overrightarrow{\nu d}(R|S) \in [\![\gamma\{g/z\}]\!]$. Now $M \in [\![\phi]\!]$ and $\overrightarrow{\nu d}(R|S) \in [\![\gamma\{g/z\}]\!]$ imply $M|\overrightarrow{\nu d}(R|S) \in [\![\phi \wr \gamma\{g/z\}]\!]$ by induction. We can assume $\vec{d} \notin fn(M)$, so we get $M|\overrightarrow{\nu d}(R|S) \equiv \overrightarrow{\nu d}(M|R|S)$ and $\overrightarrow{\nu d}(M|R|S) \in [\![\phi \wr \gamma\{g/z\}]\!]$ by Lemma 5.8(1). By Lemma 5.9(1), this implies $M|N\{\vec{e}/\vec{f}\} \in [\![\langle \bar{t}g\rangle(\phi \wr \gamma\{g/z\})]\!]$, since $M|N\{\vec{e}/\vec{f}\} \longrightarrow\!\!\!\!\rightarrow \overrightarrow{\nu d}(M|\bar{t}g.R|S)$. So we have $M|N\{\vec{e}/\vec{f}\} \in [\![\langle \bar{t}(z)\rangle(\phi \wr \gamma)]\!]$ by Theorem 5.7, since $\langle \bar{t}g\rangle(\phi \wr \gamma\{g/z\}) = \langle \bar{t}g\rangle(\phi \wr \gamma)\{g/z\} \leqslant \langle \bar{t}(z)\rangle(\phi \wr \gamma)$ by **A11**(2) (we can assume $z \neq t$ and $z \notin fn(\phi)$). We conclude by Lemma 5.10(3) that $M|N \in [\![\overrightarrow{[e=f]}\langle \bar{t}(z)\rangle(\phi \wr \gamma)]\!]$.

Case (ii). In this case $N\{\vec{e}/\vec{f}\} \longrightarrow\!\!\!\!\rightarrow \nu z\overrightarrow{\nu d}(\bar{t}z.R|S)$ and $\overrightarrow{\nu d}(R|S) \in [\![\gamma]\!]$.

Now $M \in [\![\phi]\!]$ and $\overrightarrow{\nu d}(R|S) \in [\![\gamma]\!]$ imply $M|\overrightarrow{\nu d}(R|S) \in [\![\phi \wr \gamma]\!]$ by induction. We can assume $\vec{d} \notin fn(M)$, so we get $M|\overrightarrow{\nu d}(R|S) \equiv \overrightarrow{\nu d}(M|R|S)$ and $\overrightarrow{\nu d}(M|R|S) \in [\![\phi \wr \gamma]\!]$ by Lemma 5.8(1). By Lemma 5.9(1), this implies $\overrightarrow{\nu d}(M|\bar{t}z.R|S) \in [\![\langle \bar{t}z\rangle(\phi \wr \gamma)]\!]$. Since $M|N\{\vec{e}/\vec{f}\} \longrightarrow\!\!\!\!\rightarrow \nu z\overrightarrow{\nu d}(M|\bar{t}z.R|S)$, we get $M|N\{\vec{e}/\vec{f}\} \in [\![\nu z\langle \bar{t}z\rangle(\phi \wr \gamma)]\!]$ by Lemma 5.9(3). We can now conclude by Lemma 5.10(3) that $M|N \in [\![\overrightarrow{[e=f]}\langle \bar{t}(z)\rangle(\phi \wr \gamma)]\!]$.

Proof of $M|N \in [\![\overrightarrow{[a=b]}\,\overrightarrow{[e=f]}[y=t]\nu z(\beta\{z/x\} \wr \gamma\{y/t\})]\!]$.

Case (i).

$$\begin{array}{lcll}
\overrightarrow{\nu c}(P|Q) \in [\![\beta]\!] & \Longrightarrow & \overrightarrow{\nu c}(P|Q)\{g/x\} \in [\![\beta\{g/x\}]\!] \text{ by Lemma 5.4(4)} & (\diamond) \\
\overrightarrow{\nu d}(R|S) \in [\![\gamma\{g/z\}]\!] & \Longrightarrow & \overrightarrow{\nu d}(R|S)\{y/t\} \in [\![\gamma\{g/z\}\{y/t\}]\!] \text{ by Lemma 5.4(4)} & (\diamond\diamond)
\end{array}$$

$$\begin{array}{lll}
(\diamond) \text{ and } (\diamond\diamond) & \Longrightarrow & \overrightarrow{\nu c}(P|Q)\{g/x\}|\overrightarrow{\nu d}(R|S)\{y/t\} \in [\![\beta\{g/x\} \wr \gamma\{g/z\}\{y/t\}]\!] \\
& & \text{by induction} \\
& \Longrightarrow & M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\}\{y/t\} \in [\![\beta\{g/x\} \wr \gamma\{g/z\}\{y/t\}]\!] \\
& & \text{by Lemma 5.8(1) since} \\
& & M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\}\{y/t\} \twoheadrightarrow \overrightarrow{\nu c}(P|Q)\{g/x\}|\overrightarrow{\nu d}(R|S)\{y/t\} \\
& \Longrightarrow & M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\}\{y/t\} \in [\![\nu z(\beta\{z/x\} \wr \gamma\{y/t\})]\!] \\
& & \text{by Theorem 5.7, since} \\
& & \beta\{g/x\} \wr \gamma\{g/z\}\{y/t\} = (\beta\{z/x\} \wr \gamma\{y/t\})\{g/z\} \\
& & \text{(we can assume } z \notin fn(\beta)) \ \leqslant \nu z(\beta\{z/x\} \wr \gamma\{y/t\}) \\
& & \text{by Axiom } \mathbf{A11}(2), \\
& \Longrightarrow & M|N \in [\![\overrightarrow{[a=b]}\overrightarrow{[e=f]}[y=t](\beta\{g/x\} \wr \gamma\{g/z\}\{y/t\})]\!] \\
& & \text{by Lemma 5.10(3)}
\end{array}$$

Case (*ii*).

$$\begin{array}{llll}
\overrightarrow{\nu c}(P|Q) \in [\![\beta]\!] & \Longrightarrow & \overrightarrow{\nu c}(P|Q)\{z/x\} \in [\![\beta\{z/x\}]\!] \text{ by Lemma 5.4(4)} & (\bullet) \\
\overrightarrow{\nu d}(R|S) \in [\![\gamma]\!] & \Longrightarrow & \overrightarrow{\nu d}(R|S)\{y/t\} \in [\![\gamma\{y/t\}]\!] \text{ by Lemma 5.4(4)} & (\bullet\bullet)
\end{array}$$

$$\begin{array}{lll}
(\bullet) \text{ and } (\bullet\bullet) & \Longrightarrow & \overrightarrow{\nu c}(P|Q)\{y/x\}|\overrightarrow{\nu d}(R|S)\{y/t\} \in [\![\beta\{z/x\} \wr \gamma\{y/t\}]\!] \text{ by induction} \\
& \Longrightarrow & M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\}\{y/t\} \in [\![\nu z(\beta\{z/x\} \wr \gamma\{y/t\})]\!] \\
& & \text{by Lemma 5.9(3) since} \\
& & M\{\vec{a}/\vec{b}\}|N\{\vec{e}/\vec{f}\}\{y/t\} \twoheadrightarrow \nu z(\overrightarrow{\nu c}(P|Q)\{z/x\}|\overrightarrow{\nu d}(R|S)\{y/t\}) \\
& \Longrightarrow & M|N \in [\![\overrightarrow{[a=b]}\overrightarrow{[e=f]}[y=t]\nu z(\beta\{z/x\} \wr \gamma\{y/t\})]\!] \\
& & \text{by Lemma 5.10(3).}
\end{array}$$

□

From Theorem 5.7 and Lemma 5.11, we get the main property of our realizability interpretation.

Theorem 5.12. (Soundness and completeness) $\vdash M : \sigma$ if and only if $M \in [\![\sigma]\!]$.

Proof. Soundness is proved by induction on the derivation of $\vdash M : \sigma$. The base of the induction is $\vdash M : \omega$, which is trivial. In the induction case we look at the last rule of the derivation. For Rule ($\leqslant$) we apply Theorem 5.7. For all the other rules we apply the induction hypothesis and the corresponding clause of Lemma 5.11.

As for completeness, assume that $M \in [\![\sigma]\!]$, then for all i, $1 \leqslant i \leqslant n$, $M \in [\![\phi_i]\!]$, where $\sigma \simeq \phi_1 \wedge \cdots \wedge \phi_n$ and each ϕ_i is normal. So we only have to show that if $M \in [\![\phi]\!]$, then $\vdash M : \phi$, when ϕ is normal. This proof is by structural induction on ϕ.

— If $\phi = \langle x(y)\rangle\beta$, then $M \in [\![\phi]\!]$ implies $M \twoheadrightarrow \overrightarrow{\nu c}(x(y).N|Q)$ for some $\overrightarrow{\nu c}(N|Q) \in [\![\beta]\!]$ where $x \notin \vec{c}$. We can assume that $y \notin \vec{c}$. By induction, $\vdash \overrightarrow{\nu c}(N|Q) : \beta$. This implies by Lemmas 4.7(4) and 4.7(5) that $\vdash N : \sigma$, $\vdash Q : \tau$, for some σ, τ such that $\overrightarrow{\nu c}(\sigma \wr \tau) \leqslant \beta$. So using Rule ($\langle\,\rangle$I), we have $\vdash x(y).N : \langle x(y)\rangle\sigma$, and then by Rules ($\wr$ I$_|$) and (νI), we can deduce $\vdash \overrightarrow{\nu c}(x(y).N|Q) : \overrightarrow{\nu c}(\langle x(y)\rangle\sigma \wr \tau)$. We get by Rule ($\leqslant$) that $\vdash \overrightarrow{\nu c}(x(y).N|Q) : \langle x(y)\rangle\beta$, since $\overrightarrow{\nu c}(\langle x(y)\rangle\sigma \wr \tau) \simeq \langle x(y)\rangle\overrightarrow{\nu c}(\sigma \wr \tau) \leqslant \langle x(y)\rangle\beta$ by Axiom **A5**(1) and Rule **R1**(1). By the invariance of typing under subject expansion (Theorem 4.11) we conclude $\vdash M : \langle x(y)\rangle\beta$.

— If $\phi = \langle x[z = (y)]\rangle\beta$, then we have $M \in [\![\phi]\!]$ implies $M \twoheadrightarrow \overrightarrow{\nu c}(x(y).N|Q)$ for some $\overrightarrow{\nu c}(N|Q)\{z/y\} \in [\![\beta]\!]$ where $x, z \notin \vec{c}$. We can assume that $y \notin \vec{c}$. By induction, $\vdash \overrightarrow{\nu c}(N|Q)\{z/y\} : \beta$. This implies by Lemmas 4.3(1) and 4.9(2) that $\vdash \overrightarrow{\nu c}(N|Q) : \rho$

for some ρ such that $\rho\{z/y\} \leqslant \beta$. Then by Lemmas 4.7(4) and 4.7(5), we have that $\vdash N : \sigma$, and $\vdash Q : \tau$, for some σ, τ such that $\overrightarrow{\nu c}(\sigma \wr \tau) \leqslant \rho$. As in the previous case, we can deduce $\vdash \overrightarrow{\nu c}(x(y).N|Q) : \overrightarrow{\nu c}(\langle x(y)\rangle\sigma \wr \tau)$. So, by Rule ($\leqslant$), we get that $\vdash \overrightarrow{\nu c}(x(y).N|Q) : \langle x[z = (y)]\rangle\beta$, since $\overrightarrow{\nu c}(\langle x(y)\rangle\sigma \wr \tau) \simeq \langle x(y)\rangle\overrightarrow{\nu c}(\sigma \wr \tau) \leqslant \langle x(y)\rangle\rho \leqslant \langle x[z = (y)]\rangle\rho \simeq \langle x[z = (y)]\rangle\rho\{z/y\} \leqslant \langle x[z = (y)]\rangle\beta$ by Axioms **A5**(1), **A11**(3) and **A6**(4), and Rules **R1**(1) and **R1**(3). By the invariance of typing under subject expansion (Theorem 4.11), we conclude $\vdash M : \langle x[z = (y)]\rangle\beta$.

— If $\phi = \langle\bar{x}(y)\rangle\beta$, then $M \in [\![\phi]\!]$, and $M \in [\![\phi]\!]$ implies that: either $M \twoheadrightarrow \overrightarrow{\nu c}(\bar{x}z.N|Q)$ for some $\overrightarrow{\nu c}(N|Q) \in [\![\beta\{z/y\}]\!]$, or $M \twoheadrightarrow \nu y \overrightarrow{\nu c}(\bar{x}y.N|Q)$ for some $\overrightarrow{\nu c}(N|Q) \in [\![\beta]\!]$, where $x, z \notin \vec{c}$. We can assume that $y \notin \vec{c}$.
In the first case, by induction $\vdash \overrightarrow{\nu c}(N|Q) : \beta\{z/y\}$. This implies, by Lemmas 4.7(4) and 4.7(5), $\vdash N : \sigma$, $\vdash Q : \tau$, for some σ, τ such that $\overrightarrow{\nu c}(\sigma \wr \tau) \leqslant \beta\{z/y\}$. So using Rule ($\langle\bar{\ }\rangle$ I), we have $\vdash \bar{x}z.N : \langle\bar{x}z\rangle\sigma$, and then by ($\wr$ I$_|$) and (νI), we can deduce $\vdash \overrightarrow{\nu c}(\bar{x}z.N|Q) : \overrightarrow{\nu c}(\langle\bar{x}z\rangle\sigma \wr \tau)$. Then applying Rule ($\leqslant$), we get $\vdash \overrightarrow{\nu c}(N|Q) : \langle\bar{x}(y)\rangle\beta$, since $\overrightarrow{\nu c}(\langle\bar{x}z\rangle\sigma \wr \tau) \simeq \langle\bar{x}z\rangle\overrightarrow{\nu c}(\sigma \wr \tau) \leqslant \langle\bar{x}z\rangle\beta\{z/y\} \leqslant \langle\bar{x}(y)\rangle\beta$ by Axiom **A5**(1), Rule **R1**(1) and Axiom **A11**(2). By the invariance of typing under subject expansion (Theorem 4.11), we conclude $\vdash M : \langle\bar{x}(y)\rangle\beta$.
In the second case, by induction, $\vdash \overrightarrow{\nu c}(N|Q) : \beta$. This implies, by Lemmas 4.7(4) and 4.7(5), that $\vdash N : \sigma$, $\vdash Q : \tau$, for some σ, τ such that $\overrightarrow{\nu c}(\sigma \wr \tau) \leqslant \beta$. As in the previous case, we can deduce $\vdash \overrightarrow{\nu c}(\bar{x}y.N|Q) : \overrightarrow{\nu c}(\langle\bar{x}y\rangle\sigma \wr \tau)$. So, by Rule ($\nu$I), we get that $\vdash \nu y \overrightarrow{\nu c}(\bar{x}y.N|Q) : \nu y \overrightarrow{\nu c}(\langle\bar{x}y\rangle\sigma \wr \tau)$. Therefore $\vdash \nu y \overrightarrow{\nu c}(N|Q) : \langle\bar{x}(y)\rangle\beta$ by Rule ($\leqslant$), since $\nu y \overrightarrow{\nu c}(\langle\bar{x}y\rangle\sigma \wr \tau) \simeq \langle\bar{x}(y)\rangle\overrightarrow{\nu c}(\sigma \wr \tau) \leqslant \langle\bar{x}(y)\rangle\beta$ by Axiom **A5**(1) and Rule **R1**(1). By the invariance of typing under subject expansion (Theorem 4.11), we conclude $\vdash M : \langle\bar{x}(y)\rangle\beta$.

— If $\phi = [x = y]\psi$, then $M \in [\![\phi]\!]$ implies $M\{x/y\} \in [\![\psi]\!]$. By induction, $\vdash M\{x/y\} : \psi$, which implies $\vdash M : [x = y]\psi$ by Lemma 4.9(3), taking into account that $y \notin fn(\psi)$ since $[x = y]\psi$ is normal. □

Now we are able to characterize convergency by means of typing and proving the adequacy of the filter model.

Theorem 5.13. (Resource property)
For all M:

1 $\vdash M : \langle\bar{x}(y)\rangle\omega$ for some $y \neq x$ if and only if $M \Downarrow_{\bar{x}}^{\mathrm{may}}$, and
2 $\vdash M : \langle x(y)\rangle\omega$ for some y if and only if $M \Downarrow_{x}^{\mathrm{may}}$.

Proof.

1 *'Only If'* – Suppose $\vdash M : \langle\bar{x}(y)\rangle\omega$ $(y \neq x)$. By Theorem 5.12, it follows that $M \in [\![\langle\bar{x}(y)\rangle\omega]\!]$. By Definition 5.3, we can have two cases: either $M \twoheadrightarrow \overrightarrow{\nu c}(\bar{x}z.P|Q)$ for some z, or $M \twoheadrightarrow \nu y \overrightarrow{\nu c}(\bar{x}y.P|Q)$, where $x \notin \vec{c}$. So in both cases $M \Downarrow_{\bar{x}}^{\mathrm{may}}$ since $x \notin \vec{c}$.

'If' – Suppose $M \Downarrow_{\bar{x}}^{\mathrm{may}}$ for a term M. This means that $M \twoheadrightarrow \overrightarrow{\nu c}(\bar{x}z.P|Q)$ for some $\vec{c}$, z, P, Q, where $x \notin \vec{c}$. Since $\vdash \bar{x}z.P : \langle\bar{x}z\rangle\omega$ and $\vdash Q : \omega$, we have, by Rule ($\wr$ I$_|$), that $\vdash \bar{x}z.P|Q : \langle\bar{x}z\rangle\omega \wr \omega$, which implies $\vdash \bar{x}z.P|Q : \langle\bar{x}z\rangle\omega$, since by Axiom **A2**(1) $\langle\bar{x}z\rangle\omega \wr \omega \simeq \langle\bar{x}z\rangle\omega$. Therefore $\vdash \overrightarrow{\nu c}(\bar{x}z.N|Q) : \overrightarrow{\nu c}\langle\bar{x}z\rangle\omega$ by Rule (νI). If $z \notin \vec{c}$, we have $\overrightarrow{\nu c}\langle\bar{x}z\rangle\omega \simeq \langle\bar{x}z\rangle\overrightarrow{\nu c}\,\omega \leqslant \langle\bar{x}z\rangle\omega \leqslant \nu z\,\langle\bar{x}z\rangle\omega \simeq \langle\bar{x}(y)\rangle\omega$,

by Axioms **A5**(1), **A11**(1), **A11**(2) and **A1**, where $x \neq y$. If $z \in \vec{c}$, we have $\overrightarrow{\nu c}\langle \bar{x}z\rangle\omega \simeq \nu z\, \overrightarrow{\nu d}\,\langle \bar{x}z\rangle\omega \simeq \nu z\, \langle \bar{x}z\rangle \overrightarrow{\nu d}\,\omega \leqslant \langle \bar{x}(y)\rangle\omega$, by Axioms **A3**(1), **A5**(1), **A11**(1) and **A1**, where $\vec{d} = \vec{c} - z$ and $x \neq y$. In both cases, using Rule $(\leqslant)$, we get $\vdash \overrightarrow{\nu c}\,(\bar{x}z.N|Q) : \langle \bar{x}(y)\rangle\omega$. So by Theorem 4.11, we conclude $\vdash M : \langle \bar{x}(y)\rangle\omega$.

2 The proof is similar. □

The adequacy of the filter model with respect to the open testing preorder means that the congruence induced does not consider equivalent programs that can be distinguished by some substitution and tester. We first prove the adequacy with respect to the observational preorder, the other equivalences will follow easily.

Theorem 5.14. (Adequacy)

1 If $M \sqsubseteq_F N$, then $M \sqsubseteq_O N$.
2 If $M \sqsubseteq_F N$, then $M \sqsubseteq_{OT} N$.
3 If $M \sqsubseteq_F N$, then $M \sqsubseteq_T N$.

Proof.

1 First we show that for all terms M and N,

$$\text{if } (M \sqsubseteq_F N \text{ and } M \Downarrow_{\bar{x}}^{\text{may}}),\ \text{then } N \Downarrow_{\bar{x}}^{\text{may}}\,. \qquad (*)$$

Let $M, N \in \Pi$ and assume that $M \sqsubseteq_F N$ and $M \Downarrow_{\bar{x}}^{\text{may}}$. Then $\vdash M : \langle \bar{x}(y)\rangle\omega$ for some $y \neq x$ by Theorem 5.13(1). By $M \sqsubseteq_F N$, we also have that $\vdash N : \langle \bar{x}(y)\rangle\omega$, and again by Theorem 5.13(1), we get that $N \Downarrow_{\bar{x}}^{\text{may}}$.

Moreover, note that (by Lemma 4.7) $M \sqsubseteq_F N$ implies that $C[M] \sqsubseteq_F C[N]$ for all contexts $C[\cdot]$. So, if $M \sqsubseteq_F N$, then for every context $C[\cdot]$ we have, by $(*)$, that

$$C[M] \Downarrow_{\bar{x}}^{\text{may}} \text{ implies } C[N] \Downarrow_{\bar{x}}^{\text{may}}\,.$$

Therefore $M \sqsubseteq_O N$.

2 This is an easy consequence of Point 1 and Proposition 2.8.
3 This is immediate from Point 2. □

To prove full abstraction, let us consider a normal type $\phi = \overrightarrow{[a = b]}\beta$, where β is a body. We associate to each body β a test term $T_{\beta,w}$, where w is a name that does not occur in ϕ. Then, to discriminate using type ϕ, we will equate channels according to $\overrightarrow{[a = b]}$ and put the obtained process in parallel with $T_{\beta,w}$.

The test terms are defined by induction on bodies: they offer the communications 'complementary' to the prefixes of the bodies. The test terms look similar to the test bodies (Definition 3.10(1)). Notably, the test bodies turn out to be the principal types of the test terms.

Definition 5.15. Let β be a body and w be a name that does not occur in β. We define the test term $T_{\beta,w}$ by structural induction on β:

1 $T_{\omega,w} = \nu v\, \bar{w}v.\mathbf{0}$
2 $T_{\langle x(y)\rangle\gamma,w} = \nu y\, \bar{x}y.T_{\gamma,w}$
3 $T_{\langle x[z=(y)]\rangle\gamma,w} = \bar{x}z.T_{\gamma,w}$

4 $T_{\langle \bar{x}y\rangle\gamma,w} = x(z).[y = z]T_{\gamma,w}$ where $z \notin fn(\langle \bar{x}y\rangle\gamma)$
5 $T_{\langle \bar{x}(y)\rangle\gamma,w} = x(y).T_{\gamma,w}$.

From Proposition 4.5, each $T_{\beta,w}$ has a principal type, since ! does not occur in it. More precisely, we have the following proposition.

Proposition 5.16. Let β be a body and w be a name that does not occur in β. Then $\wp(T_{\beta,w}) = \Im_{\beta,w}$.

Proof. By structural induction on β, using the definition of principal type (Definition 4.4). □

The following property of $\Im_{\beta,w}$ is crucial for our proof.

Lemma 5.17. Let β be a body, σ be a type, and w be a name such that $w \notin fn(\beta)$, $w \notin fn(\sigma)$. Then $\sigma \wr \Im_{\beta,w} \leqslant \langle \bar{w}(v)\rangle\omega$ if and only if $\sigma \leqslant \beta$.

Proof. 'if' – $\sigma \leqslant \beta$ implies $\sigma \wr \Im_{\beta,w} \leqslant \beta \wr \Im_{\beta,w}$ by Rule **R1**(4). Moreover, $\beta \wr \Im_{\beta,w} \leqslant \langle \bar{w}(v)\rangle\omega$ can be easily proved by induction on β.
'Only If' – Assume that

$$\sigma \wr \Im_{\beta,w} \leqslant \langle \bar{w}(v)\rangle\omega \text{ and } \sigma \nleqslant \beta.$$

By Definition 3.10, $\Gamma_{\beta,w}[\cdot] \simeq [\cdot] \wr \Im_{\beta,w}$. Let $\sigma \simeq \psi_1 \wedge \cdots \wedge \psi_n$, where $\psi_1, \cdots, \psi_n$ are normal types. If $\sigma \nleqslant \beta$ then for all $i \leqslant n$, $\psi_i \nleqslant^{\mathsf{N}} \beta$, by Theorem 3.14. Then, by Lemma 3.11(1), we have $\overrightarrow{\nu c}\Gamma_{\beta,w}[\psi_i] \simeq \omega$ for all $i \leqslant n$, where $\vec{c} = \bigcup_{i \leqslant n} fn(\psi_i) \cup fn(\beta)$. From $\sigma \wr \Im_{\beta,w} \leqslant \langle \bar{w}(v)\rangle\omega$, we get $\Gamma_{\beta,w}[\sigma] \simeq \bigwedge_{i \leqslant n} \Gamma_{\beta,w}[\psi_i] \leqslant \langle \bar{w}(v)\rangle\omega$ and, also, $\overrightarrow{\nu c}(\bigwedge_{i \leqslant n} \Gamma_{\beta,w}[\psi_i]) \leqslant \overrightarrow{\nu c}\langle \bar{w}(v)\rangle\omega \leqslant \langle \bar{w}(v)\rangle\omega$ by Rule **R1**(2) and Axioms **A5**(1) and **A11**(1). So we get a contradiction. □

Lemma 5.18. Let $\phi = \overrightarrow{[a = b]}\beta$ be a normal type, where β is a body. Then $\vdash M : \phi$ if and only if $\vdash M\{\vec{a}/\vec{b}\} : \beta$.

Proof. Observe that $\beta \simeq \phi\{\vec{a}/\vec{b}\}$. The 'Only If' follows from Lemma 4.9(1), and the 'if' from Lemma 4.9(3). □

Theorem 5.19. (Characterization theorem) Let $\phi = \overrightarrow{[a = b]}\beta$ be a normal type, where β is a body, and let w be a name that does not occur in ϕ.

1 Let N be a process that does not contain w. Then $N|T_{\beta,w} \Downarrow^{\mathrm{may}}_{\bar{w}}$ if and only if $\vdash N : \beta$.
2 Let M be a process that does not contain w. Then $M\{\vec{a}/\vec{b}\}|T_{\beta,w} \Downarrow^{\mathrm{may}}_{\bar{w}}$ if and only if $\vdash M : \phi$.

Proof.

1 By Theorem 5.13, $N|T_{\beta,w} \Downarrow^{\mathrm{may}}_{\bar{w}}$ if and only if $\vdash N|T_{\beta,w} : \langle \bar{w}(y)\rangle\omega$. By Lemma 4.7(4), we get $\vdash N : \sigma$ and $\vdash T_{\beta,w} : \tau$ for some σ, τ such that $\sigma \wr \tau \leqslant \langle \bar{w}(y)\rangle\omega$. By Propositions 4.5 and 5.16, $\vdash T_{\beta,w} : \tau$ implies $\Im_{\beta,w} \leqslant \tau$. So we get $\sigma \wr \Im_{\beta,w} \leqslant \langle \bar{w}(y)\rangle\omega$ by Rule **R1**(4). By Lemma 5.17 and Rule ($\leqslant$), we can conclude that $\vdash N : \beta$.
2 This is immediate from Point 1 and Lemma 5.18. □

Now we can prove our full abstraction result.

Theorem 5.20. (Full abstraction)

1 $M \sqsubseteq_{OT} N$ implies $M \sqsubseteq_F N$.
2 Let M be a body process. Then $M \sqsubseteq_T N$ implies $M \sqsubseteq_F N$.

Proof.

1 Assume that $M \not\sqsubseteq_F N$, say $\vdash M : \sigma$ and $\nvdash N : \sigma$ for some type σ. Then by Proposition 3.7 and Rule ($\leqslant$), there is a normal type $\phi = \overrightarrow{[a = b]}\beta$, where β is a body, such that $\vdash M : \phi$ and $\nvdash N : \phi$. We have, by Theorem 5.19(2), that $M\{\vec{a}/\vec{b}\}|T_{\beta,w} \Downarrow_{\bar{w}}^{\text{may}}$ but not $N\{\vec{a}/\vec{b}\}|T_{\beta,w} \Downarrow_{\bar{w}}^{\text{may}}$ (where w is a name not occurring in M, N). So we can conclude that $M \not\sqsubseteq_{OT} N$.

2 Let $\vdash M : \sigma$ and $\nvdash N : \sigma$ be as in the proof of Point 1. By Proposition 4.3(1), we can find $\tau \leqslant \sigma$ such that $\vdash_{-\leqslant} M : \tau$. Clearly $\nvdash N : \tau$. By Definition 4.13, $\tau \simeq \gamma_1 \wedge \cdots \wedge \gamma_n$ ($n \geqslant 1$) where $\gamma_1, \ldots, \gamma_n$ are body types, since M is a body process. So there is a body $\beta \in \{\gamma_1, \ldots, \gamma_n\}$ such that $\vdash M : \beta$ and $\nvdash N : \beta$. Then Theorem 5.19(1) implies that $M|T_{\beta,w} \Downarrow_{\bar{w}}^{\text{may}}$ holds but $N|T_{\beta,w} \Downarrow_{\bar{w}}^{\text{may}}$ does not hold. So we can conclude that $M \not\sqsubseteq_T N$.

$\square$

As an immediate consequence, we have that the open testing preorder and the observational preorder coincide. Moreover, for body processes, open testing coincides with testing.

Corollary 5.21.

1 $M \sqsubseteq_{OT} N$ if and only if $M \sqsubseteq_O N$.

2 Let M be a body process. Then $M \sqsubseteq_{OT} N$ if and only if $M \sqsubseteq_T N$.

For example, let $P = \bar{x}y.\mathbf{0}$ and $Q = \nu y\, \bar{x}y.\mathbf{0}$. It it easy to see that $Q \sqsubseteq_F P$ and $P \not\sqsubseteq_F Q$. In fact, for $\phi \equiv \langle \bar{x}y \rangle \omega$, we have $\vdash P : \phi$ and $\nvdash Q : \phi$.
Now, according to Theorem 5.20, we can build the test term $T_{\phi,w}$ and the substitution s such that $s(P)|T_{\phi,w} \Downarrow_{\bar{w}}^{\text{may}}$ and not $s(Q)|T_{\phi,w} \Downarrow_{\bar{w}}^{\text{may}}$, where w is a name not occurring in P, Q. Indeed, $T_{\phi,w} = x(z).[y = z]\nu v \bar{w}v.\mathbf{0}$ and the empty substitution are the required test term and substitution.

Remark 5.22. Corollary 5.21 shows that $\sqsubseteq_{OT}$ is a congruence, and clearly $\sqsubseteq_{OT}$ implies $\sqsubseteq_T$. As a matter of fact, $\sqsubseteq_{OT}$ is the biggest congruence included in $\sqsubseteq_T$. To see this, take M, N such that $M \not\sqsubseteq_{OT} N$, but $M \sqsubseteq_T N$. By definition, there is a context $C[\cdot]$ such that $C[M] \not\sqsubseteq_T C[N]$, so we cannot extend $\sqsubseteq_{OT}$ preserving the closure under contexts.

6. Related papers

The literature related to the present paper was partially quoted in the introduction. We add here comparisons and remarks that require the development of the filter model presented in previous sections.

Two papers closely related to our own are Hennessy (1991) and Boreale and De Nicola (1995). Both papers describe denotational semantics of π-calculus that are fully abstract with respect to a 'must' operational semantics. The models described are term models and the completeness proofs rely on the existence of head normal forms. The language considered in Hennessy (1991) is quite expressive, since it includes an if–then–else construct. The fully abstract model is built using a proof system based on a set of inequations between processes. The compact elements of the domain turn out to be the finite processes that have a 'finite' behaviour on input actions. Also, the language of

Boreale and De Nicola (1995) is richer than ours since it includes mismatch. The axioms and rules of Boreale and De Nicola (1995) for comparing finite processes are a subset of our axioms and rules for comparing types (modulo the correspondence between types and processes). There is also a similarity between the strong normal forms of Boreale and De Nicola (1995) and our normal types.

The main feature of our approach in comparison with the one of Boreale and De Nicola (1995) is that the meanings of $\alpha.P$ and νxP, that is the set of types that can be assigned to $\alpha.P$ and νxP, are defined from the meaning of P in a uniform way. For instance if $[\![P]\!]$ is the filter of types that can be assigned to P, $[\![\nu xP]\!] = \{\sigma \mid \nu x\tau \leqslant \sigma \text{ for some } \tau \in [\![P]\!]\}$. In Boreale and De Nicola (1995), instead, the interpretation of prefixes and restrictions is done, by using, in a suitable way, the equivalence class of the strong normal form of the term itself. For instance, $[\![\nu xP]\!] = [snf(\nu xP)]$, where $snf(\nu xP)$ is the strong normal form of νxP. So the definition of the model in Boreale and De Nicola (1995) requires strong normal forms. Our normal types are only a tool for proving adequacy and full abstraction of the filter model.

A number of interpretations of the π-calculus based on functor categories have been proposed (Stark 1996; Fiore *et al.* 1996; Hennessy 1996). Such interpretations are parameterized on the set of names used by a process. Let $\mathscr{I}$ be the category of finite sets of names with injections. The functor category considered for the interpretation is the category of functors from $\mathscr{I}$ to the category of **SFP** in Stark (1996) and the category of functors from $\mathscr{I}$ to **Cpo** (and **Set**) in Fiore *et al.* (1996). In such interpretations we have to consider the meaning of processes indexed by a set of names that may change as the process does input actions or restrictions. The bound output is modelled by

$$N \times (N \multimap P),$$

where N is the functor injecting $\mathscr{I}$ into the chosen category (**SFP** or **Cpo**). The functor $N \multimap P$ is such that

$$(N \multimap P)s = P(s + \mathbf{1})$$

where s is a finite set of names and $s + \mathbf{1}$ is obtained by adding a new name to s. (In the category of finite sets with injections, $s + \mathbf{1}$ does not depend on which element we add.) The fact that $\multimap$ is a functor implies that the interpretation of the bound output does not depend on the name chosen (the name is fresh). The input is modelled by

$$N \times (N \to P),$$

where the functor $\to$ is such that

$$(N \to P)s = (P\ s)^s \times P(s + \mathbf{1}).$$

$(P\ s)^s$ specifies the behaviour of the function for inputs in s, and $P(s + \mathbf{1})$ specifies the value of the function when the input is not in s. In this second case, as for the $\multimap$ functor, the result is independent of the particular element chosen. In these models we can interpret faithfully restriction and input, and we can express the fact that the restriction operator introduces a new name that should not interfere with the free names of other processes.

The models presented in Fiore *et al.* (1996) and Stark (1996) are shown to be fully abstract with respect to strong late bisimilarity. A similar technique is used in Hennessy (1996) to present two fully abstract term models for the 'may' and 'must' testing. It is also outlined that the model for the 'may' testing is also fully abstract with respect to the more general equivalence that compares the 'may' behaviour of processes in all contexts.

None of these functorial models, however, has a logical presentation. The full abstraction of the model in Hennessy (1996), which is the closest to ours, is provided similarly to Hennessy (1988) by associating both to terms and to elements of the domain the sets of *acceptances* (similar to strong normal forms). In the first case the set of acceptances of a term determines its operational behaviour, and in the second the set of acceptances of an element of the domain completely determines the element (it is the set of its finite approximations). So, for Hennessy's work, acceptances are a device to prove completeness (and not essential to the definition of the model), just as normal types are for our work. A drawback of the functorial interpretation, which was already noticed for the interpretation of block structures for which the functorial approach was first introduced in Oles (1985), is that it is not clear how to extend it to higher-types.

Hartonas and Hennessy (1997) presents a model of a subset of the language FACILE (mini-FACILE) that is fully abstract with respect to the 'may' observational preorder that compares the behaviour of terms in all contexts. The technique used for the proof of full abstraction is similar to that of the present paper. That is, the logic of the model is specified through a type assignment system. Moreover, a notion of satisfaction for types that coincide with our realizability is given. The type assignment provides a correct and complete proof system for the 'may' properties expressed by the types. The language modelled, mini-FACILE, is a typed λ-calculus with a type proc for processes that are specified with a value passing CCS-like syntax. The part of the language modelled does not, however, contain the local scoping operator (which corresponds to the π-calculus restriction operator), and adding static scoping is left as future work. It is hinted that static scoping could be modelled via functor categories. Even though papers such as the previously quoted Stark (1996), Fiore *et al.* (1996) and Hennessy (1996) show that this is the case, the difficulty with these interpretations is to get a logical characterization of the domains through a type assignment system. Indeed, at the moment we do not know of anyone who has succeeded in this. We instead model static scoping via the inclusion (and equivalence) relation between types.

Acknowledgments

We thank Rocco De Nicola, Eugenio Moggi and Michele Boreale for enlightening discussions about the subject of this paper, and Andrea Pederiva for his comments on drafts of this paper.

The present version of this paper has strongly benefitted from comments and remarks by the anonymous referees.

Mariangiola Dezani-Ciancaglini was visiting the Tokyo Institute of Technology during the elaboration of the final version of this paper and she would like to thank her host

Masako Takahashi-Horai and the whole Department of Mathematical and Computing Sciences for the ideal working conditions they provided.

References

Abramsky, S. (1991a) Domain Theory in Logical Form. *Annals of Pure and Applied Logic* **51** 1–77.

Abramsky, S. (1991b) A Domain Equation for Bisimulation. *Information and Computation* **92** (2) 161–218.

Abramsky, S. and Ong, C.-H. L. (1993) Full Abstraction in the Lazy Lambda Calculus. *Information and Computation* **105** (2) 159–267.

Amadio, R. M. and Dam, M. (1996) Toward a modal theory of types for the π-calculus. In: Formal Techniques in Real-Time and Fault-Tolerant Systems. *Springer-Verlag Lecture Notes in Computer Science* **1135** 347–365.

Barendregt, H. P. (1984) *The λ-Calculus — Its Syntax and Semantics*, North-Holland.

Barendregt, H. P., Coppo, M. and Dezani-Ciancaglini, M. (1983) A Filter Lambda Model and the Completeness of Type Assignment. *Journal of Symbolic Logic* **48** (4) 931–940.

Boreale M. and De Nicola R. (1995) Testing Equivalence for Mobile Processes. *Information and Computation* **120** (2) 279–303.

Boreale M. and Sangiorgi D. (1996) Some Congruence Properties for π-calculus Bisimilarities. Technical Report RR-2870, INRIA-Sophia Antipolis.

Boudol G. (1994) A Lambda Calculus for (Strict) Parallel Functions. *Information and Computation* **108** (1) 51–127.

Dam M. (1993) Model Checking Mobile Processes. In: CONCUR'93. *Springer-Verlag Lecture Notes in Computer Science* **715** 22–36.

De Nicola R. and Hennessy M. (1984) Testing Equivalences for Processes. *Theoretical Computer Science* **34** (1–2) 83–133.

De Nicola R. and Hennessy M. (1987) CCS without τ's. In: TAPSOFT'87. *Springer-Verlag Lecture Notes in Computer Science* **249** 138–152.

Dezani-Ciancaglini M., de'Liguoro U. and Piperno A. (1996) Filter Models for Conjunctive-Disjunctive λ-calculi. *Theoretical Computer Science* **170**(1–2) 83–128.

Dezani-Ciancaglini M., de'Liguoro U. and Piperno A. (1998) A Filter Model for Concurrent λ-calculi. *SIAM Journal on Computing* **27** (5) 1376–1419.

Fiore M. P., Moggi E. and Sangiorgi D. (1996) A Fully Abstract Model for the π-calculus. In: *LICS'96*, IEEE Comp. Soc. Press.

Hartonas T. and M. Hennessy M. (1997) Full Abstractness for a Functional/Concurrent Language with Higher-Order Value-Passing. Computer Science Technical Report 97:01, School of Cognitive and Computing Sciences, University of Sussex. (Extended Abstract in *CSL'97*.) *Information and Computation* (to appear).

Hennessy M.(1988) *Algebraic Theory of Processes*, MIT Press.

Hennessy M.(1991) A Model for the π-calculus. Computer Science Technical Report 91:08, School of Cognitive and Computing Sciences, University of Sussex.

Hennessy M.(1994a) A Fully Abstract Denotational Model for Higher-Order Processes. *Information and Computation* **112** (1) 55–95.

Hennessy M.(1994b) Higher-order Processes and their Models. In: ICALP '94. *Springer-Verlag Lecture Notes in Computer Science* **820** 286–303.

Hennessy M.(1996) A Fully Abstract Denotational Semantics for the π-calculus. Computer Science Technical Report 96:04, School of Cognitive and Computing Sciences, University of Sussex.

Milner R. (1980) A Calculus of Communicating Systems. *Springer-Verlag Lecture Notes in Computer Science* **92**.

Milner R. (1989) *Communication and Concurrency*, Prentice-Hall.

Milner R. (1991) The Polyadic π-calculus: A Tutorial. Technical Report ECS-LFCS-91-180, Edinburgh Univ. (Also in *Logic and Algebra of Specification*, Springer-Verlag, Berlin, 1993.)

Milner R. (1992) Functions as Processes. *Mathematical Structures in Computer Science* **2**(2) 119–141.

Milner R., Parrow J. and Walker D. (1992) A Calculus of Mobile Processes, I and II. *Information and Computation* **100** (1) 1–77.

Milner R., Parrow J. and Walker D. (1993) Modal Logics for Mobile Processes. *Theoretical Computer Science* **114** (1) 149–171.

Oles F. J. (1985) Type Algebras, Functor Categories and Block Structures. *Algebraic Methods in Semantics*, Cambridge University Press 543–573.

Ong C.-H. L. (1993) Non-Determinism in a Functional Setting. In: *LICS '93*, IEEE Comp. Soc. Press 275–286.

Sangiorgi D. (1992) Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms, Ph. D. Thesis, University of Edinburgh.

Sangiorgi D. (1996) A Theory of Bisimulation for the π-calculus. *Acta Informatica* **33** (1) 68–97.

Scott D. (1982) Domains for Denotational Semantics. In: ICALP'82. *Springer-Verlag Lecture Notes in Computer Science* **140** 577–613.

Stark I. (1996) A Fully Abstract Domain Model for the π-calculus. In: *LICS'96*, IEEE Comp. Soc. Press 36–42.

Thomsen B. (1990) Calculi for Higher-Order Communicating Systems, Ph. D. Thesis, Imperial College.