The Responsive Bisimulations in the polar π-calculus

—— Achieving Equivalence when Equivalent Objects are not Equivalent

UNSW-CSE-TR-0203

Xiaogang Zhang and John Potter
School of Computer Science and Engineering
University of New South Wales, Australia
{xzhang,potter}@cse.unsw.edu.au

Abstract

Ongoing work attempts to model concurrent object systems using process algebra. The behaviour of an object can be described as the composition of a process representing the basic functionality of the object and separated processes controlling the concurrent behaviour of that object. However, familiar bisimulations, including the weak barbed equivalence, are too strong to capture the behavioural equivalence between object components. This paper proposes the responsive bisimulation, an even weaker bisimulation relation which considers that delaying an incoming message locally has the same effect as delaying it externally, as long as potential interference by competing receptors is avoided. With this bisimulation, an equivalence between the π-calculus expression $(\nu n)(m.\overline{n} \mid k.n.P)$ *and k.m.P then can be achieved. The responsive bisimulation is congruence for the family of processes which model objects.*

The Responsive Bisimulations in the polar π-calculus
—— Achieving Equivalence when Equivalent Objects are not Equivalent

(Revised Version)

Xiaogang Zhang and John Potter
School of Computer Science and Engineering
University of New South Wales, Australia
{xzhang,potter}@cse.unsw.edu.au

Abstract

Ongoing work attempts to model concurrent object systems using process algebra. The behaviour of an object can be described as the composition of a process representing the basic functionality of the object and separated processes controlling the concurrent behaviour of that object. However, familiar bisimulations, including the weak barbed equivalence, are too strong to capture the behavioural equivalence between object components. This paper proposes the responsive bisimulation, an even weaker bisimulation relation which considers that delaying an incoming message locally has the same effect as delaying it externally, as long as potential interference by competing receptors is avoided. With this bisimulation, an equivalence between the π-calculus expression $(\nu n)(m.\overline{n} \mid k.n.P)$ *and* $k.m.P$ *then can be achieved. The responsive bisimulation is congruence for the family of processes which model objects.*

1 Introduction

With the ability to directly model dynamic reference structures, process algebra such as the π-calculus ([Milner92], [Milner99]) and its variations have been applied to modelling concurrent object systems ([Walker95], [Jones93], [Sangiorgi96], [Hüttel96], [Zhang97]). Some researchers ([Schneider97], [Zhang98A], [Zhang98B]) have also applied it in modelling compositional concurrent objects in the aspect-oriented programming style ([Aksit92], [Holmes97]) to avoid the inheritance anomaly [McHale94].

One of the important issues in these object modelling endeavours is to identify the similarity between some composed behaviours and the expected behaviour. There are many known bisimulation techniques available for various purposes. For example, the weak ground bisimulation and many others can recognise the equality between processes $(\nu n)(m.\overline{n} \mid n.P)$ and $m.P$, by ignoring the internal forwarding. However, those familiar bisimulations can fail for some behaviours, such as $(\nu n)(m.\overline{n} \mid k.n.P)$ and $k.m.P$, which we want to be equalised for compositional objects. The necessary of this kind equivalence can be shown by the following "real world" communication example:

> In the mailroom of a business skyscraper, the property manager uses internal mail to send bills to her tenants and collect payments. Each tenant has a locked mailbox, which located either on the mailroom wall and can be opened from outside of the mailroom by the tenant, or on the door of the tenant's suite and a postman delivers mails from mailroom to the tenant's suite. There are a couple of techniques the manager may adopt to classify the behaviour of a tenant. Most traditional bisimulations require the manager to monitor everything around the mailroom, including when each item of mail is taken away. Therefore, whether the mailbox is at the mailroom or not will make difference. Even with the barbed equivalence (its definition will be shown in section 3), for which the manager needs only to examine which mails (both incoming and outgoing) are in the mailroom, the mailbox's location still will make a difference for those tenants who lost their key: for some the bill remains in mailroom, while for others the bill has gone. In fact, the manager is not interested in those details at all; she only wants to

know about the arrival of payments and classify tenants who pay the bill on time, and in cash, as behaving “good”, and so on.

To describe the problem a little bit more formally, lets review the idea of [Zhang98A] and [Zhang98B] in modelling concurrent objects in the π-calculus. The behaviour of a concurrent object can be modelled as the parallel composition of two processes: a process F which represents the object’s functional behaviour and can be expressed with the generic form $F \stackrel{\text{def}}{=} \prod !n_i(\tilde{x}).M_i\langle\tilde{x}\rangle$, and a process C which represents the constraints on the object’s concurrent behaviour. In effect, F on its own, represents an object with no constraints on its concurrent interactions. For example, the functionality of a buffer object can be described by the expression $F_B \stackrel{\text{def}}{=} !n_r(x).M_r\langle x\rangle \mid !n_w(x).M_w\langle x\rangle$, where $n_r(x).M_r\langle x\rangle$ and $n_w(x).M_w\langle x\rangle$ represent the behaviour of the read and write methods respectively, each of them can have unlimited invocations executing in parallel without any concern of interfering among them. To discipline those invocations, assume a synchronisation behaviour modelled by the control process $C_s \stackrel{\text{def}}{=} m_r(x).\overline{n_r}\langle x\rangle + m_w(x).\overline{n_w}\langle x\rangle$, where the sum operator in fact represents a mutual exclusion lock on those methods. Then the parallel composition of the two processes, $(\nu\, n)(C_s \mid F)$, will be weakly bisimilar to $R_s \stackrel{\text{def}}{=} m_r(x).M_r\langle x\rangle + m_w(x).M_w\langle x\rangle$, as expected. More complicated and generic method exclusion relations, besides mutual exclusion locks, can be simply modelled and composed in exactly the same way, once the κ-calculus ([Zhang01A]) is used. The κ-calculus is an extended calculus which welds the mobility power of the π-calculus with the synchronisation expressiveness of the algebra of exclusion ([Noble00]).

Let the process O_1 and O_2 illustrated in Figure 1-1 represent two different versions of the internal structure of the same composed object in a state where its only method is blocked by the lock of key κ. The only difference between them is that O_1 has an extra “empty” control $Ctrl_e$ (the postman) which does nothing but forwards whatever message received from channel m to the next control $Ctrl_l$ (the locked mailbox). The body (a tenant) of these two can always give the same response (a payment) if fed with the same message (a bill). If an unlocking signal is received via channel κ, both O_1 and O_2 can accept incoming messages and process them immediately. If some message arrives before the unlocking, O_1 will store it in an internal buffer (the door mailbox) and delay the process until unlocked, but O_2 will leave the message in the external buffer (the mailroom) as it was, while waiting for unlocking.

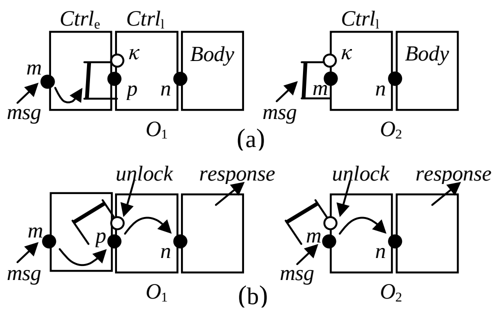


Figure 1-1

For a client (the manager) who is sending the message, the behaviour of the target object can be measured only by observing how it responds. Therefore, the behaviour of O_1 and O_2 are identical in the client's eyes, since the responses they can give are the same (both from the same *Body*). However, this behavioural similarity cannot be captured by most of the known behavioural equivalence relations, since in some stage O_1 can perform an input action from the channel m while O_2 cannot. Even the weak barbed-equivalence, one of the weakest, is too strong for them, since $O_1 \mid R$ and $O_2 \mid R$ are not weakly barbed-bisimilar for some R, such as $R \stackrel{\text{def}}{=} \overline{m}\langle a\rangle$.

In this paper we propose the notion of responsive bisimulation to recover this kind of equivalence. The basic idea can come up from two different viewpoints. One view is, when testing the behaviour of a process, to “localise” each test message that was sent by some client and buffered in the environment, so that it can only be accessible by the target process, but not be visible to the observer and mistaken as an output from the target process. Another view is, when determining the similarity between process evolution trees, the delay of incoming messages at the input end is tolerant. In section 3 we will show these two views are equivalent.

One of the major results of this paper is that, the responsive bisimulation is a congruence for the family of processes which model objects. This is the consequence of the fact that replication preserves the responsive bisimulation for this family of processes. The preservation is also held by parallel composition for an even larger category of processes, and held by other operations for all situations.

Another interesting result, revealed in Proposition 3-22, is that, a persistently available receptor with an internal forwarder can be ignored.

With ability to derive the equivalence of a larger collection of behaviours, the responsive bisimulation can capture the similarity of responsive behaviours of object processes, and more interestingly, the general behaviour of control processes. As one of the major significance, it permits a deep study on the properties of object composition. For example, lets denote $C \succsim F$ for the operation composing an object component process F with a control process C, a special kind of object component process, to yield a new object component process with expected behaviour. With the responsive bisimulation relation, we not only have the associative law, i.e., $C_1 \succsim (C_2 \succsim F) \equiv (C_1 \succsim C_2) \succsim F$, but also the identity law, i.e., there is some empty control (identity) E such that for all F composible with E, the composed object $E \succsim F$ is equivalent to the original object F, and for all control process C compatible with E, the three control processes $E \succsim C$, $C \succsim E$ and C are all equivalent ([Zhang01C]).

In this paper the responsive bisimulation is presented in the polar π-calculus, an asynchronous π-calculus with polars. This allows us presenting the features essential for responsive bisimulation, without the full complexity of the κ-calculus. Its extension to the κ-calculus will be studied further in [Zhang01B].

The rest of the paper is structured as follows: section 2 briefly introduces the polar π-calculus and related notions; section 3 defines responsive bisimulation; section 4 gives some properties of the equivalence and other theoretical results; section 5 discusses some further issues relating the responsive bisimulation with other notions; section 6 briefly describes some applications of responsive bisimulation in modelling compositional objects with related results; and section 7 concludes the paper.

2 The polar π-calculus (π_p-calculus)

The *polar π-calculus* (π_p-calculus) can be considered as a subcalculus of the asynchronous π-calculus ([Amadio96] and [Hüttel96]), with the restriction that for any input guarded term $m(\tilde{x}).P$, in P no free occurrence of a name in $\tilde{x}$ can be used as the channel name (subject) of an input action. This restriction is enforced syntactically by introducing the input and output polars of a name.

The asynchronous π-calculus (π_a-calculus) itself, as pointed out in [Amadio96], is a subcalculus of the standard π-calculus, with the restriction that outputs cannot be used as prefix or as a choice point. Names in the π_a-calculus have no polarity, and can be transmitted through communication channels for receivers to use in either input actions or output actions. The same name m can be used as the subject of either an output action $\overline{m}\langle\tilde{u}\rangle$ or an input action $m(\tilde{u})$, distinguished by the presence or absence of an overbar. When an output and an input action with the same name as the subject can take place in parallel, then a communication may be committed.

Polarised names were introduced by [Odersky95a], where each name has either an input or output polarity, both can be transmitted in communication, and also both can be used as subject of any action (with a rule that a term will be inactive if the polarity is wrong, that is, when a name with input polarity is used as subject of output term, or a name with output polarity used as subject of input term). Since both polarities can be transmitted, the matching operator, for testing name identity and guaranteeing the desired commitment, has to involve a pair of names with the same identity but opposite polarities.

In our polar π-calculus, just as in the π_a-calculus, output is non-blocking and is not used as prefix. And similar to [Odersky95a], in the polar π-calculus a name m, that can be considered as a reference to a communication channel, has two polars, the input polar $\overset{+}{m}$ and the output polar $\overline{m}$, which can be considered as the input port from, and the output port to, respectively, the channel m. The main difference from [Amadio96] and [Odersky95a] is that, in the polar π-calculus, only output polar of names can be transmitted through a communication channel. [Ravara97] and others have adopted a similar restriction, but in the polar π-calculus this restriction is enforced syntactically. As a consequence, only output polar substitution can be caused by input prefix, while that in [Odersky95a] may involve names with both polarities and in [Amadio96] will affect both input and output usage of a name.

One of the advantages in using polars to enforce this restriction rather than using the implicit restriction as that in [Ravara97] and [Merro00], we believe, is the simplicity and clearness in describing and proving some properties of bisimulations, such as when expressing a bisimilarity between process P and Q being maintained between $m(x).P$ and $m(x).Q$.

The notion of polarised ports is a base for many forms of communications, including postal mail, email and telephone. It also providers a base for semantics of message passing and the object-oriented computation model. The following scenario illustrates communication over polarised ports:

> A new email account was established for agent A, with a mailbox $\overset{+}{e}_A$ for A to receive emails and an email address $\overset{-}{e}_A$ for A to give to other people. The first mail A found in this mailbox was a greeting message which included the system administrator's email address $\overset{-}{e}_S$. Then A sent a mail to $\overset{-}{e}_S$ asking about agent B's email address, and the reply was "$\overset{-}{e}_B$". Then A sent a message to $\overset{-}{e}_B$ saying "My email address is $\overset{-}{e}_A$, I have a confidential document for you", and got a reply "Please send the document to my another email address $\overset{-}{e}_{B2}$".

It is easy to see from this scenario, the input polar of a name, such as mailbox $\overset{+}{e}_A$, cannot be sent; the output polar, $\overset{-}{e}_A$, of the same channel can be transmitted, but the receiver cannot use it for receiving messages. It is not only making no sense for A to tell B "Here is an email address $\overset{+}{e}_{A2}$, you should use it to receive my next email", but also means that the behaviour of B would depend on whether $\overset{+}{e}_{A2}$ is a valid emailbox, which B should not be responsible for. In other words, if input polars can be transmitted, then an agent's behaviour will no longer be predictable from its structure. This issue can be demonstrated more intuitively when the κ-calculus is introduced ([Zhang01A],[Zhang01B]), where an input port can be locked/unlocked, and a change of an input name may also change the locking status.

For modelling object system, the prohibition of transmitting input polars can be also described as "the ownership of an input port (or reference) cannot be transferred". We will see the need of this again later.

Another important treatment in the polar π-calculus is that the silent action τ becomes a derived action and restricted to be internal. Section 2.2 will give detailed description and discussion about this issue (see rule tr_INTL, Remark 2-3 and Notation 2-6).

2.1 The syntax of the polar π-calculus

Let $\mathcal{N}$ be the set of all names, and ranged over by name expressions m,n,u,v and variables x,y. Let $\overset{+}{\mathcal{N}} \overset{\text{def}}{=} \{\overset{+}{n}: n \in \mathcal{N}\}$ and $\overset{-}{\mathcal{N}} \overset{\text{def}}{=} \{\overset{-}{n}: n \in \mathcal{N}\}$ be the sets of input polars and output polars respectively. Let polar expressions a,b and variable w range over the set of all polars, $\overset{+}{\mathcal{N}} \cup \overset{-}{\mathcal{N}}$. Both $\tilde{r}$ and $\{r_{i\in I}\}$, where I is an index set of arity n, are abbreviations for $r_1,r_2,\ldots,r_n$. The generic process terms P in the polar π-calculus are generated by the following grammars:

$$P ::= \overset{-}{m}\langle\tilde{\overset{-}{u}}\rangle \,\big|\, (\nu\,\tilde{n})P \,\big|\, P_1 \,|\, P_2 \,\big|\, !B \,\big|\, G \,\big|\, A\langle\tilde{a}\rangle, \qquad G ::= \mathbf{0} \,\big|\, B \,\big|\, (\nu\,\tilde{n})G \,\big|\, G_1 + G_2, \qquad B ::= \overset{+}{m}(\tilde{\overset{-}{x}}).P$$

The set of all actions a process may take is specified by $\alpha ::= \overset{+}{m}(\tilde{\overset{-}{u}}) \,\big|\, (\nu\tilde{v})\overset{-}{m}\langle\tilde{\overset{-}{u}}\rangle \,\big|\, \tau$, where $\tilde{v} \subseteq \tilde{u}$ and $m \notin \tilde{v}$.

Here $\mathbf{0}$ is the inactive (terminated) process; $\overset{-}{m}\langle\tilde{\overset{-}{u}}\rangle$ is the output action which sends output polars $\tilde{\overset{-}{u}}$ into the channel m; $(\nu\,\tilde{n})P$ binds the set of names $\tilde{n}$, and therefore both polars of each of those names, within the scope of P; $P_1 \,|\, P_2$ indicates two processes run in parallel; $A\langle\tilde{a}\rangle$ is an instance of parameterised process agent; giving the process agent abstraction $A \overset{\text{def}}{=} (\tilde{w})P$ is obeying $((\tilde{w})P)\langle\tilde{a}\rangle \equiv P\{\tilde{a}/\tilde{w}\}$; B is an input-guarded process; $!B$ is the replication and G is the exclusive choice, both have to be constructed from input-guarded processes.

> **Notation 2−1**: As usual, we need auxiliary functions fn, bn and n to identify the sets of free, bound and all names, respectively, of a term or action. As a calculus with polars, we also need more specified functions to identify polars. For process term, we define:

$$in(\overset{-}{m}\langle\tilde{\overset{-}{u}}\rangle) \overset{\text{def}}{=} \varnothing; \qquad in((\nu\tilde{n})P) \overset{\text{def}}{=} \{\tilde{n}\} \cup in(P); \qquad in(P_1 \,|\, P_2) \overset{\text{def}}{=} in(P_1) \cup in(P_2);$$

$$in(!B) \overset{\text{def}}{=} in(B); \qquad in(\overset{+}{m}(\tilde{\overset{-}{x}}).P) \overset{\text{def}}{=} \{m\} \cup in(P); \qquad in(G_1 + G_2) \overset{\text{def}}{=} in(G_1) \cup in(G_2);$$

$$in((\tilde{\overset{+}{u}},\tilde{\overset{-}{v}})P) \overset{\text{def}}{=} \{\tilde{u}\} \cup in(P);$$

$$bin(\overset{-}{m}\langle\tilde{\overset{-}{u}}\rangle) \overset{\text{def}}{=} \varnothing; \qquad bin((\nu\tilde{n})P) \overset{\text{def}}{=} \{\tilde{n}\} \cup bin(P); \qquad bin(P_1 \,|\, P_2) \overset{\text{def}}{=} bin(P_1) \cup bin(P_2);$$

$bin(!B) \stackrel{def}{=} bin(B)$; $\quad bin({}^{+}m(\tilde{x}).P) \stackrel{def}{=} bin(P)$; $\quad bin(G_1+G_2) \stackrel{def}{=} bin(G_1) \cup bin(G_2)$;
$bin(({}^{+}\tilde{u},{}^{-}\tilde{v})P) \stackrel{def}{=} \{\tilde{u}\} \cup bin(P)$;

$on({}^{-}m\langle\tilde{u}\rangle) \stackrel{def}{=} \{m,\tilde{u}\}$; $\quad on((\nu\tilde{n})P) \stackrel{def}{=} \{\tilde{n}\} \cup on(P)$; $\quad on(P_1 \mid P_2) \stackrel{def}{=} on(P_1) \cup on(P_2)$;
$on(!B) \stackrel{def}{=} on(B)$; $\quad on({}^{+}m(\tilde{x}).P) \stackrel{def}{=} \{\tilde{x}\} \cup on(P)$; $\quad on(G_1+G_2) \stackrel{def}{=} on(G_1) \cup on(G_2)$;
$on(({}^{+}\tilde{u},{}^{-}\tilde{v})P) \stackrel{def}{=} \{\tilde{v}\} \cup on(P)$;

$bon({}^{-}m\langle\tilde{u}\rangle) \stackrel{def}{=} \varnothing$; $\quad bon((\nu\tilde{n})P) \stackrel{def}{=} \{\tilde{n}\} \cup bon(P)$; $\quad bon(P_1 \mid P_2) \stackrel{def}{=} bon(P_1) \cup bon(P_2)$;
$bon(!B) \stackrel{def}{=} bon(B)$; $\quad bon({}^{+}m(\tilde{x}).P) \stackrel{def}{=} \{\tilde{x}\} \cup bon(P)$; $\quad bon(G_1+G_2) \stackrel{def}{=} bon(G_1) \cup bon(G_2)$;
$bon(({}^{+}\tilde{u},{}^{-}\tilde{v})P) \stackrel{def}{=} \{\tilde{u}\} \cup bon(P)$;

For actions, we define:

$in((\nu\tilde{v}){}^{-}m\langle\tilde{u}\rangle) \stackrel{def}{=} \{\tilde{v}\}$; $\quad in({}^{+}m(\tilde{u})) \stackrel{def}{=} \{m\}$; $\quad in(\tau) \stackrel{def}{=} \varnothing$;
$bin((\nu\tilde{v}){}^{-}m\langle\tilde{u}\rangle) \stackrel{def}{=} \{\tilde{v}\}$; $\quad bin({}^{+}m(\tilde{u})) \stackrel{def}{=} \varnothing$; $\quad bin(\tau) \stackrel{def}{=} \varnothing$;
$on((\nu\tilde{v}){}^{-}m\langle\tilde{u}\rangle) \stackrel{def}{=} \{m\} \cup \{\tilde{v}\} \cup \{\tilde{u}\}$; $\quad on({}^{+}m(\tilde{u})) \stackrel{def}{=} \{\tilde{u}\}$; $\quad on(\tau) \stackrel{def}{=} \varnothing$;
$bon((\nu\tilde{v}){}^{-}m\langle\tilde{u}\rangle) \stackrel{def}{=} \{m\} \cup (\{\tilde{v}\} \cap \{\tilde{u}\})$; $\quad bon({}^{+}m(\tilde{u})) \stackrel{def}{=} \varnothing$; $\quad bon(\tau) \stackrel{def}{=} \varnothing$;

And for both P terms and actions, we define

$fin(t) \stackrel{def}{=} in(t) - bin(t)$; $\quad fon(t) \stackrel{def}{=} on(t) - bon(t)$;
$fn(t) \stackrel{def}{=} fin(t) \cup fon(t)$; $\quad bn(t) \stackrel{def}{=} bin(t) \cup bon(t)$; $\quad n(t) \stackrel{def}{=} in(t) \cup on(t) \equiv fn(t) \cup bn(t)$.

Notation 2−2: The generic form of output actions $(\nu\tilde{v}){}^{-}m\langle\tilde{u}\rangle$ may be abbreviated as ${}^{-}m\langle\tilde{u}\rangle$ when $\tilde{v}=\varnothing$, as ${}^{-}m(\tilde{u})$ when $\tilde{v}=\tilde{u}$; and as ${}^{-}m$ when $\tilde{u}=\varnothing$ or $\tilde{u}$ is not of interest. The input guarded term ${}^{+}m(\tilde{x}).P$ may be abbreviated as ${}^{+}m.\,P$ when $\tilde{x}=\varnothing$ or $\tilde{x}$ is not of interest. The standard abbreviation $\prod P_i \stackrel{def}{=} P_1 \mid P_2 \mid \ldots \mid P_2$ and $\sum G_i \stackrel{def}{=} G_1+G_2+\ldots+G_3$ are also used throughout this paper.

2.2 The semantics of the polar π-calculus

The structural equivalences and labelled transitions are shown in Figure 2-1 and Figure 2-2.

Remark 2−3:Rule tr_INTL gives the meaning of the internal action τ. It can be equivalently written as: If $P \xrightarrow{(\nu\tilde{v}){}^{-}m\langle\tilde{u}\rangle} P'$ and $Q \xrightarrow{{}^{+}m(\tilde{u})} Q'$ where $\tilde{v} \cap fn(Q)=\varnothing$, then $(\nu m)(P \mid Q) \xrightarrow{\tau} (\nu m)(\nu\tilde{v})(P' \mid Q')$.

In rule tr_INTL, the name restriction (νm) over communication channel is necessary for preserving internal actions in name substitution, and therefore preserving any bisimulation involving τ action, under the input prefixing ${}^{+}m(\tilde{x}).P$. In a non-polarised π-calculus, if a process P is able to perform a synchronisation action, then for any input prefixing $m(\tilde{x})$ and names $\tilde{u}$, after the transition $m(\tilde{x}).P \xrightarrow{m(\tilde{u})} P\{\tilde{u}/\tilde{x}\}$, the process $P\{\tilde{u}/\tilde{x}\}$ can always be able to perform a synchronisation action, since both polars of a name will be substituted. But this will not be true for the polar π-calculus, where the input prefixing ${}^{+}m(\tilde{x}).P$ will cause a substitution only on the output polars, and without the name restriction the ability of P to perform an "internal" action can be altered by such a substitution, and as the consequence, the oτ-bisimulation will not be preserved by input prefixing. The argument here is, only when a communication take places via a channel which is not visible nor interruptible from external observers, then it can be considered as a true internal action. It is only from this point of view, that the standard rule $fn(\tau)=bn(\tau)=\varnothing$ ([Amadio96]) can make sense.

Definition 2−4: As usual, let ( )* indicate the contents in ( ) repeating zero or finitely many times, then the weak transitions are defined as: $P \stackrel{\tau}{\Longrightarrow} P'$ iff $P(\xrightarrow{\tau})^{*}P'$; $\quad P \stackrel{\alpha}{\Longrightarrow} P'$ iff $P \stackrel{\tau}{\Longrightarrow} . \xrightarrow{\alpha} . \stackrel{\tau}{\Longrightarrow} P'$, where $\alpha \neq \tau$.

Reduction relation, a familiar concept in this literature, is defined in a non-standard way in the π_p-calculus:

Definition 2−5: $P \to P'$ iff $(\nu m)P \xrightarrow{\tau} (\nu m)P'$ for some m; $\quad P \Longrightarrow P'$ iff $(\nu m)P \stackrel{\tau}{\Longrightarrow} (\nu m)P'$ for some m.

Summation		
str-SUM1:	$P_1 \mid \mathbf{0} \equiv P_1$;	$G_1 + \mathbf{0} \equiv G_1$
str-SUM2:	$P_1 \mid P_2 \equiv P_2 \mid P_1$;	$G_1 + G_2 \equiv G_2 + G_1$
str-SUM3:	$P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$;	$G_1 + (G_2 + G_3) \equiv (G_1 + G_2) + G_3$
Scope		
str-SCP1:	$(\nu\, \tilde{n})P \equiv P$, if $\tilde{n} \cap fn(P) = \varnothing$;	$(\nu\, \tilde{n})\, G \equiv G$, if $\tilde{n} \cap fn(G) = \varnothing$
str-SCP2:	$(\nu\, m)\, \bar{m}\langle \tilde{y} \rangle \equiv \mathbf{0}$;	$(\nu\, m)\, \overset{+}{m}(\tilde{x}).P \equiv \mathbf{0}$
str-SCP3:	$(\nu\, \tilde{m})(\nu\, \tilde{n})P \equiv (\nu\, \tilde{m}, \tilde{n})P$;	$(\nu\, \tilde{m})(\nu\, \tilde{n})P \equiv (\nu\, \tilde{n})(\nu \tilde{m})P$
str-SCP4:	$(\nu\, \tilde{n}) P_1 \mid P_2 \equiv (\nu\, \tilde{n})(P_1 \mid P_2)$, if $\tilde{n} \cap fn(P_2) = \varnothing$;	$(\nu\, \tilde{n}) G_1 + G_2 \equiv (\nu\, \tilde{n})(G_1 + G_2)$, if $\tilde{n} \cap fn(G_2) = \varnothing$
str-REN:	$(\nu\, \tilde{n}) P \equiv (\nu\, \tilde{m})(P\{\tilde{m}/\tilde{n}\})$, if $\tilde{m} \cap fn(P) = \varnothing$	
Instance		
str-INST:	$((\tilde{w})P)\langle \tilde{a} \rangle \equiv P\{\tilde{a}/\tilde{w}\}$	

Figure 2-1 Structural congruence rules for the polar π-calculus

tr_OUT: $$\frac{\cdot}{\bar{m}\langle \tilde{u} \rangle \xrightarrow{\bar{m}\langle \tilde{u} \rangle} \mathbf{0}_{\mathbf{P}}}, \qquad \frac{P \xrightarrow{\bar{m}\langle \tilde{u} \rangle} P', \quad m \notin \tilde{v}}{(\nu \tilde{v})P \xrightarrow{(\nu\, \tilde{v})\bar{m}\langle \tilde{u} \rangle} P'}$$

tr_IN: $$\frac{\cdot}{\overset{+}{m}(\tilde{x}).P \xrightarrow{\overset{+}{m}(\tilde{u})} P\{\tilde{u}/\tilde{x}\}}$$

tr_RES: $$\frac{P \xrightarrow{\alpha} P', \quad \tilde{n} \cap fn(\alpha) = \varnothing}{(\nu \tilde{n})P \xrightarrow{\alpha} (\nu \tilde{n})P'}$$

tr_REP: $$\frac{B \xrightarrow{\overset{+}{m}(\tilde{u})} P}{!B \xrightarrow{\overset{+}{m}(\tilde{u})} P \mid !B}$$

tr_PARL: $$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

tr_CHOI: $$\frac{G_1 \xrightarrow{\overset{+}{m}(\tilde{u})} P}{G_1 + G_2 \xrightarrow{\overset{+}{m}(\tilde{u})} P}$$

tr_INTL: $$\frac{P \xrightarrow{(\nu\, \tilde{v})\bar{m}\langle \tilde{u} \rangle} P', \quad P \xrightarrow{\overset{+}{m}(\tilde{u})} P''}{(\nu\, m)P \xrightarrow{\tau} (\nu\, m)(\nu \tilde{v})P''}$$

tr_STRUC: $$\frac{P_1' \equiv P_1, \quad P_1 \xrightarrow{\alpha} P_2, \quad P_2 \equiv P_2'}{P_1' \xrightarrow{\alpha} P_2'}$$

Figure 2-2 Labelled transition rules for process terms in the polar π-calculus

Clearly, $P \xrightarrow{\tau} P'$ implies $P \longrightarrow P'$, and $P \overset{\tau}{\Longrightarrow} P'$ implies $P \Rightarrow P'$ and, therefore, a variant of the rule tr_INTL can be written as: if $P \xrightarrow{(\nu\, \tilde{v})\bar{m}\langle \tilde{u} \rangle} P'$ and $Q \xrightarrow{\overset{+}{m}(\tilde{u})} Q'$ where $\tilde{v} \cap fn(Q) = \varnothing$, then $P \mid Q \longrightarrow (\nu \tilde{v})\, (P' \mid Q')$. Besides the reason we have just discussed, the distinction between internal action and reduction is also necessary for the new bisimulation relation, and we will find out later.

The τ action does not appear as a guard of the B term in the syntax of the polar π-calculus, but can be treated as a derived notation.

Notation 2−6: τ-guarded processes are abbreviations defined as follows, where $m \notin fn(P)$ and $m \notin fn(G)$:

$$\tau.P \overset{\text{def}}{=} (\nu\, m)\, (\overset{+}{m}.P \mid \bar{m}); \qquad \tau.P + G \overset{\text{def}}{=} (\nu\, m)\, (\, (\overset{+}{m}.P + G) \mid \bar{m}); \qquad !\tau.P \overset{\text{def}}{=} (\nu\, m)\, (\overset{+}{m}.(P \mid !\tau.P) \mid \bar{m})$$

Definition 2−7: The strong commitments of process P are defined as:

P can *commit* the action α, denoted as $P{\downarrow}\alpha$, if there exists some P' such that $P \xrightarrow{\alpha} P'$.

P can *commit* on input polar $\overset{+}{m}$, denoted as $P{\downarrow}\overset{+}{m}$, if there exists some input action $\alpha = \overset{+}{m}(\tilde{u})$ s.t. $P{\downarrow}\alpha$;

P can *commit* on output polar $\bar{m}$, denoted as $P{\downarrow}\bar{m}$, if there is some output action $\alpha = (\nu \tilde{v})\bar{m}\langle \tilde{u} \rangle$ s.t. $P{\downarrow}\alpha$;

P can *commit* the action sequence $\ell = \alpha_1, \alpha_2, \ldots, \alpha_n$, denoted as $P{\downarrow}\ell$, if $P \xrightarrow{\alpha_1} . \xrightarrow{\alpha_2} . \ldots . \xrightarrow{\alpha_n} P'$, or $P \xrightarrow{\ell} P'$ for short.

The weak commitments $\Downarrow$, are obtained by replacing $\rightarrow$ with $\Rightarrow$ and $\downarrow$ with $\Downarrow$ throughout.

Definition 2−8: Process P' is a *derivative* of P, if there exists some finite sequence ℓ such that $P \xrightarrow{\ell} P'$.

2.3 Mapping between the π_p-calculus and π_a-calculus

From π_p to π_a

$$
\begin{array}{ll}
\llbracket \mathbf{0} \rrbracket_{p\to a} & \stackrel{\text{def}}{=} \mathbf{0}; \\
\llbracket (\nu\, n)P \rrbracket_{p\to a} & \stackrel{\text{def}}{=} (\nu\, n_i, n_o)\,(\llbracket P \rrbracket_{p\to a} \,|\, !n_o(\tilde{x}).\overline{n_i}\langle \tilde{x} \rangle); \\
\llbracket \overset{-}{n}\langle \overset{-}{\tilde{u}} \rangle \rrbracket_{p\to a} & \stackrel{\text{def}}{=} \overline{n_o}\langle \tilde{u}_o \rangle; \\
\llbracket \overset{+}{n}(\overset{-}{\tilde{x}}).P \rrbracket_{p\to a} & \stackrel{\text{def}}{=} n_i(\tilde{x}_o).\llbracket P \rrbracket_{p\to a}; \\
\llbracket P_1 \,|\, P_2 \rrbracket_{p\to a} & \stackrel{\text{def}}{=} \llbracket P_1 \rrbracket_{p\to a} \,|\, \llbracket P_2 \rrbracket_{p\to a}; \\
\llbracket !B \rrbracket_{p\to a} & \stackrel{\text{def}}{=} !\llbracket B \rrbracket_{p\to a}; \\
\llbracket G_1 + G_2 \rrbracket_{p\to a} & \stackrel{\text{def}}{=} \llbracket G_1 \rrbracket_{p\to a} + \llbracket G_2 \rrbracket_{p\to a}. \\
\llbracket A\langle \overset{+}{\tilde{u}}, \overset{-}{\tilde{v}} \rangle \rrbracket_{p\to a} & \stackrel{\text{def}}{=} \llbracket A \rrbracket_{p\to a}\langle \tilde{u}_i, \tilde{v}_o \rangle; \\
\llbracket (\overset{+}{\tilde{x}}, \overset{-}{\tilde{y}})P \rrbracket_{p\to a} & \stackrel{\text{def}}{=} (\tilde{x}_i, \tilde{y}_o)\, \llbracket P \rrbracket_{p\to a};
\end{array}
$$

From π_a to π_p

$$
\begin{array}{ll}
\llbracket \mathbf{0} \rrbracket_{a\to p} & \stackrel{\text{def}}{=} \mathbf{0}; \\
\llbracket (\nu\, n)P \rrbracket_{a\to p} & \stackrel{\text{def}}{=} (\nu\, n_i, n_o)\,(\llbracket P \rrbracket_{a\to p} \,|\, !\overset{+}{n}_o(\overset{-}{\tilde{x}}).\overset{-}{n}_i\langle \overset{-}{\tilde{x}} \rangle); \\
\llbracket \overline{n}\langle \tilde{u} \rangle \rrbracket_{a\to p} & \stackrel{\text{def}}{=} \overset{-}{n}_o\langle \overset{-}{\tilde{u}}_i, \overset{-}{\tilde{u}}_o \rangle; \\
\llbracket n(\tilde{x}).P \rrbracket_{a\to p} & \stackrel{\text{def}}{=} \overset{+}{n}_i(\overset{-}{\tilde{x}}_i, \overset{-}{\tilde{x}}_o).\, \llbracket P \rrbracket_{a\to p}; \\
\llbracket P_1 \,|\, P_2 \rrbracket_{a\to p} & \stackrel{\text{def}}{=} \llbracket P_1 \rrbracket_{a\to p} \,|\, \llbracket P_2 \rrbracket_{a\to p}; \\
\llbracket !B \rrbracket_{a\to p} & \stackrel{\text{def}}{=} !\llbracket B \rrbracket_{a\to p}; \\
\llbracket G_1 + G_2 \rrbracket_{a\to p} & \stackrel{\text{def}}{=} \llbracket G_1 \rrbracket_{a\to p} + \llbracket G_2 \rrbracket_{a\to p}; \\
\llbracket A\langle \tilde{u} \rangle \rrbracket_{a\to p} & \stackrel{\text{def}}{=} \llbracket A \rrbracket_{a\to p}\langle \overset{+}{\tilde{u}}_i, \overset{-}{\tilde{u}}_i, \overset{+}{\tilde{u}}_o, \overset{-}{\tilde{u}}_o \rangle; \\
\llbracket (\tilde{x})P \rrbracket_{a\to p} & \stackrel{\text{def}}{=} (\overset{+}{\tilde{x}}_i, \overset{-}{\tilde{x}}_i, \overset{+}{\tilde{x}}_o, \overset{-}{\tilde{x}}_o)\, \llbracket P \rrbracket_{a\to p};
\end{array}
$$

Figure 2-3 Mutual encoding of the π_a-calculus and π_p-calculus

If ignore the polar subscripts, then the π_p to π_a mapping will be exactly the same as the direct mapping, except the second clause, which should become $\llbracket (\nu\, n)P \rrbracket_{p\to a} \stackrel{\text{def}}{=} (\nu\, n)\, \llbracket P \rrbracket_{p\to a}$ in the direct mapping.

Since the polar π-calculus is a subcalculus of the asynchronous π-calculus, generic properties concluded from the full domain of the asynchronous π-calculus can also apply to the polar π-calculus in its restricted domain. For example, the ground bisimulation, early, late and open bisimulations all coincide in the asynchronous π-calculus, as well as in the polar π-calculus. The asynchronous π-calculus itself, as pointed out in [Amadio96], is a subcalculus of the standard π-calculus, with the restrictions that outputs cannot be used as prefix or on a choice point. Therefore, generic properties concluded from the standard π-calculus can also apply to the asynchronous π-calculus and the polar π-calculus.

The π_p-calculus is also a subcalculus of [Odersky95a]'s polarised π-calculus (we denote it as π_o), and all terms in the π_p can be directly converted into the π_o.

From π_p to π_o

$$
\begin{array}{ll}
\llbracket \mathbf{0} \rrbracket_{p\to o} & \stackrel{\text{def}}{=} \mathbf{0}; \\
\llbracket (\nu\, n)P \rrbracket_{p\to o} & \stackrel{\text{def}}{=} (\nu\, n)\, \llbracket P \rrbracket_{p\to o}; \\
\llbracket \overset{-}{n}\langle \overset{-}{\tilde{u}} \rangle \rrbracket_{p\to o} & \stackrel{\text{def}}{=} n!\langle \tilde{u}! \rangle; \\
\llbracket \overset{+}{n}(\overset{-}{\tilde{x}}).P \rrbracket_{p\to o} & \stackrel{\text{def}}{=} n?(\tilde{x}!).\llbracket P \rrbracket_{p\to o}; \\
\llbracket P_1 \,|\, P_2 \rrbracket_{p\to o} & \stackrel{\text{def}}{=} \llbracket P_1 \rrbracket_{p\to o} \,|\, \llbracket P_2 \rrbracket_{p\to o}; \\
\llbracket !B \rrbracket_{p\to o} & \stackrel{\text{def}}{=} *\llbracket B \rrbracket_{p\to o}; \\
\llbracket G_1 + G_2 \rrbracket_{p\to o} & \stackrel{\text{def}}{=} \llbracket G_1 \rrbracket_{p\to o} + \llbracket G_2 \rrbracket_{p\to o}. \\
\llbracket A\langle \overset{+}{\tilde{u}}, \overset{-}{\tilde{v}} \rangle \rrbracket_{p\to o} & \stackrel{\text{def}}{=} \llbracket A \rrbracket_{p\to o}\langle \tilde{u}?, \tilde{v}! \rangle; \\
\llbracket (\overset{+}{\tilde{x}}, \overset{-}{\tilde{y}})P \rrbracket_{p\to o} & \stackrel{\text{def}}{=} (\tilde{x}?, \tilde{y}!)\, \llbracket P \rrbracket_{p\to o};
\end{array}
$$

From π_o to π_p

$$
\begin{array}{llll}
\llbracket \mathbf{0} \rrbracket_{o\to p} & \stackrel{\text{def}}{=} \mathbf{0}; & & \\
\llbracket (\nu\, n)P \rrbracket_{o\to p} & \stackrel{\text{def}}{=} (\nu\, n_i, n_o)\,(\llbracket P \rrbracket_{o\to p} \,|\, !\overset{+}{n}_o(\overset{-}{\tilde{x}}).\overset{-}{n}_i\langle \overset{-}{\tilde{x}} \rangle); & & \\
\llbracket n!\langle \tilde{u} \rangle \rrbracket_{o\to p} & \stackrel{\text{def}}{=} \overset{-}{n}_o\langle \overset{-}{\tilde{u}}_i, \overset{-}{\tilde{u}}_o \rangle; & \llbracket n?\langle \tilde{u} \rangle \rrbracket_{o\to p} & \stackrel{\text{def}}{=} \mathbf{0}; \\
\llbracket n?(\tilde{x}?, \tilde{y}!).P \rrbracket_{o\to p} & \stackrel{\text{def}}{=} \overset{+}{n}_i(\overset{-}{\tilde{x}}_i, \overset{-}{\tilde{y}}_o).\, \llbracket P \rrbracket_{o\to p}; & \llbracket n!(\tilde{x}?, \tilde{y}!).P \rrbracket_{o\to p} & \stackrel{\text{def}}{=} \mathbf{0}; \\
\llbracket P_1 \,|\, P_2 \rrbracket_{o\to p} & \stackrel{\text{def}}{=} \llbracket P_1 \rrbracket_{o\to p} \,|\, \llbracket P_2 \rrbracket_{o\to p}; & & \\
\llbracket *B \rrbracket_{o\to p} & \stackrel{\text{def}}{=} !\llbracket B \rrbracket_{o\to p}; & & \\
\llbracket G_1 + G_2 \rrbracket_{o\to p} & \stackrel{\text{def}}{=} \llbracket G_1 \rrbracket_{o\to p} + \llbracket G_2 \rrbracket_{o\to p}; & & \\
\llbracket A\langle \tilde{u} \rangle \rrbracket_{o\to p} & \stackrel{\text{def}}{=} \llbracket A \rrbracket_{o\to p}\langle \overset{+}{\tilde{u}}_i, \overset{-}{\tilde{u}}_i, \overset{+}{\tilde{u}}_o, \overset{-}{\tilde{u}}_o \rangle; & & \\
\llbracket (\tilde{x})P \rrbracket_{o\to p} & \stackrel{\text{def}}{=} (\overset{+}{\tilde{x}}_i, \overset{-}{\tilde{x}}_i, \overset{+}{\tilde{x}}_o, \overset{-}{\tilde{x}}_o)\, \llbracket P \rrbracket_{o\to p}; & &
\end{array}
$$

Figure 2-4 Mutual encoding of the π_o-calculus and π_p-calculus

If ignore the polar subscripts, then the π_p to π_a mapping will be exactly the same as the direct mapping, except the second clause, which should become $\llbracket (\nu\, n)P \rrbracket_{p\to a} \stackrel{\text{def}}{=} (\nu\, n)\, \llbracket P \rrbracket_{p\to a}$ in the direct mapping.

Since the polar π-calculus is a subcalculus of the asynchronous π-calculus, generic properties concluded from the full domain of the asynchronous π-calculus can also apply to the polar π-calculus in its restricted domain. For example, the ground bisimulation, early, late and open bisimulations all coincide in the asynchronous π-calculus, as well as in the polar π-calculus. The asynchronous π-calculus itself, as pointed out in [Amadio96], is a subcalculus of the standard π-calculus, with the restrictions that outputs cannot be used as prefix or on a choice point. Therefore, generic properties concluded from the standard π-calculus can also apply to the asynchronous π-calculus and the polar π-calculus. However, most results of this paper for the polar π-calculus have been established independently of the correspondence with the asynchronous π-calculus.

3 Responsive bisimulation in the polar π-calculus

The barbed bisimulation ([Milner92b],[Sangiorgi92b]) is a rather weak relation, which traces the state changes of a process during the course of reductions and observes which channels are available for communication. We adopt the version of [Amadio96] for an asynchronous π-calculus.

> **Definition 3−9**: A symmetric relation $\mathcal{S}$ on P-terms is a (strong) **barbed bisimulation** if whenever $P\mathcal{S}Q$ then $P\downarrow a$ implies $Q\downarrow a$ for $a\in\mathcal{N}$, and $P\rightarrow P'$ implies $\exists Q'$ such that $Q\rightarrow Q'$ and $P'\mathcal{S}Q'$.
>
> Let $\sim_b$ be the largest strong barbed bisimulation. The notion of weak barbed bisimulation $\approx_b$ is obtained by replacing the transition $\downarrow$ with $\Downarrow$, and $\rightarrow$ with $\Rightarrow$ throughout.

However, since barbed bisimulation cannot identify what messages are communicated, it is too rough to measure process's behaviour. Better measurements are needed.

> **Definition 3−10**: The **process context** $\mathcal{C}[.]$ is given by $\mathcal{C}::=[.]\,|\,(\nu\tilde{n})\mathcal{C}\,|\,\mathcal{C}|P\,|\,!\dagger m(\tilde{x}).\mathcal{C}\,|\,\dagger m(\tilde{x}).\mathcal{C}+G.$
> From this syntax, the hole [.] occurs at most once in a process context expression. By filling this hole in $\mathcal{C}[.]$ with the process Q, $\mathcal{C}[Q]$ constructs a new process expression.

> **Definition 3−11**: Let $\mathcal{C}[.]$ be a process context, the strong and weak **barbed congruences** are defined as $P\simeq_b Q$ if $\forall\mathcal{C}[.].(\mathcal{C}[P]\sim_b\mathcal{C}[Q])$; $P\cong_b Q$ if $\forall\mathcal{C}[.].(\mathcal{C}[P]\approx_b\mathcal{C}[Q])$;
> and the weaker versions similar to [Amadio96]: let R be an arbitrary process, the strong and weak **barbed equivalence** are defined as: $P\simeq_{b1}Q$ if $\forall R.(R|P\sim_b R|Q)$; $P\cong_{b1}Q$ if $\forall R.(R|P\approx_b R|P)$.

Weak barbed equivalence is too strong for compositional objects, as illustrated by the example in Figure 1-1, where O_1 and O_2, the two different versions of the same object component, can be expressed in the polar π-calculus as $O_1\overset{\text{def}}{=}(\nu\,p)(!\dagger m(\tilde{y}).\bar{p}\langle\tilde{y}\rangle\,|\,(\nu\,n)(\dagger k.!\dagger p(\tilde{y}).\bar{n}\langle\tilde{y}\rangle\,|\,!\dagger n(\tilde{x}).Body))$ and $O_2\overset{\text{def}}{=}(\nu\,n)(\dagger k.!\dagger m(\tilde{y}).\bar{n}\langle\tilde{y}\rangle\,|\,!\dagger n(\tilde{x}).Body)$. If only output actions are detectable, then within an environment where there is no other place that the input polar of the same channel m is used, the behaviour of O_1 and O_2 can be considered as the same by an external observer. But this similarity of the observation behaviours cannot be captured by the weak barbed equivalence. The weak barbed equivalence fails in at least two ways.

First, it cannot distinguish between a message sent out from the target process and a message sent by another agent to the target process but buffered in the environment. For example, given the message $\bar{m}\langle\tilde{p}\rangle$, then $O_1|\bar{m}\langle\tilde{p}\rangle\Longrightarrow Q_1$ and $O_2|\bar{m}\langle\tilde{p}\rangle\Longrightarrow Q_2$, where $Q_1\overset{\text{def}}{=}(\nu\,s)(!\dagger m(\tilde{y}).\bar{s}\langle\tilde{y}\rangle\,|\,(\nu\,n)(\dagger k.!\dagger s(\tilde{y}).\bar{n}\langle\tilde{y}\rangle\,|\,!\dagger n(\tilde{x}).Body)\,|\,\bar{s}\langle\tilde{p}\rangle)$ and $Q_2\overset{\text{def}}{=}O_2|\bar{m}\langle\tilde{p}\rangle$. Since Q_1 has entered an undetectable status, while in Q_2 the message $\bar{m}\langle\tilde{p}\rangle$ remains to be a detectable "output action"; that is, $Q_1\not\Downarrow\bar{m}$ but $Q_2\Downarrow\bar{m}$, therefore $O_1|\bar{m}\langle\tilde{p}\rangle\not\approx_b O_2|\bar{m}\langle\tilde{p}\rangle$, that is, $O_1\not\cong_b O_2$.

$$
\begin{tikzcd}
 & O_1|R \arrow[dl, "\tau"'] \arrow[dr, "\tau"] & & O_2|R \arrow[d, "\tau"] \\
O_1|\bar{p}\langle\tilde{q}\rangle & & Q_1|\dagger m(x).\bar{x}\langle\tilde{q}\rangle & O_2|\bar{p}\langle\tilde{q}\rangle
\end{tikzcd}
$$

Figure 3-1

Second, it cannot prevent input names clash between the testing environment and the processes being tested. For example, let $R\overset{\text{def}}{=}\dagger m(\bar{x}).\bar{x}\langle\tilde{q}\rangle\,|\,\bar{m}\langle\tilde{p}\rangle$, then, as shown in Figure 3-1, $O_1|R$ can take two different reduction paths, either

$O_1|R \Longrightarrow O_1|\bar{p}\langle \bar{q}\rangle$ or $O_1|R \Longrightarrow Q_1|\dagger m(\bar{x}).\bar{x}\langle \bar{q}\rangle$, while $O_2|R$ has only one reduction path, $O_2|R \Longrightarrow O_2|\bar{p}\langle \bar{q}\rangle$. Therefore $O_1|R \not\approx_b O_2|R$, that is $O_1 \not\cong_b O_2$.

Another failure in the strong version is, the barbed bisimulation treats synchronisation actions occurred in public channels as single step reduction, and therefore cannot match them with uncompleted synchronisations which have delays on inputting side.

We need a different technique to measure the observation behaviours, weak enough to ignore the unrelated information and strong enough to capture the similarity in responses perceived by outsiders. As with barbed bisimulation, we must note the state changes of a process caused by internal actions, and we must also be able to detect which communication channels are available for output in all evolved states. Moreover, in order to distinguish states, we need to be able to observe what each of the messages output by the process is. The οτ-bisimulation, similar to that in [Amadio96], can provide this degree of observation:

> **Definition 3−12**: The *οτ*-**bisimulation** is a symmetric relation $\mathcal{S}$ on processes, for which whenever $P\mathcal{S}Q$ then $P\xrightarrow{\alpha}P'$ implies $Q\xrightarrow{\alpha}Q'$ and $P'\mathcal{S}Q'$, where α is a non-input action and $bn(\alpha)\cap fn(Q)=\emptyset$.
>
> The weak οτ-bisimulation is obtained by replacing $\xrightarrow{\alpha}$ with $\xRightarrow{\alpha}$ throughout. We denote the largest (strong) οτ-bisimulation as $\sim_{o\tau}$, and the largest weak οτ-bisimulation as $\approx_{o\tau}$.

Lemma 3−13: Every οτ-bisimulation $\mathcal{S}$ is preserved by restriction, i.e., $P\mathcal{S}Q$ implies $(\nu\,\tilde{n})P\mathcal{S}(\nu\,\tilde{n})Q$.

Proof: This can be proven by show that $\mathcal{R} \stackrel{\text{def}}{=} \{((\nu\,\tilde{n})P, (\nu\,\tilde{n})Q) : P\mathcal{S}Q\}$ is a $\mathcal{S}$. Here we only give that for the strong case $\mathcal{S}\subseteq\sim_{o\tau}$, the weak case can be proven similarly.

As normal in π-calculi, for each $\tilde{n}$, P and Q combination, we always assume that the rule str-REN is automatically and implicitly applied over the fresh names $\tilde{n}$ to avoid name clash. For example, assume $P \stackrel{\text{def}}{=} (\nu\, n)(A\langle \dagger n, \bar{n}\rangle \,|\, (\nu\, n)P_1)$, then a name $m \notin fn(P_1|Q)$ will be picked up automatically and the expression $(\nu\, n)(A\langle \dagger n, \bar{n}\rangle \,|\, (\nu\, n)P_1)$ will be treated as $(\nu\, n)(A\langle \dagger n, \bar{n}\rangle \,|\, (\nu\, m)P_1\{\tilde{m}/\tilde{n}\})$ implicitly without mention.

Assume $(\nu\,\tilde{n})P\xrightarrow{\alpha}P'$ for some arbitrary non-input action α, by inducting over transition rules, this is only possible in one of the following two cases:

1. $\tilde{n}\cap fn(\alpha)=\emptyset$ and $P\xrightarrow{\alpha}P''$. By rule tr_RES, $(\nu\,\tilde{n})P\xrightarrow{\alpha}(\nu\,\tilde{n})P''$, so $P'\equiv(\nu\,\tilde{n})P''$. By $P\mathcal{S}Q$, we have $Q\xrightarrow{\alpha}Q''$ and $P''\mathcal{S}Q''$. By tr_RES, $(\nu\,\tilde{n})Q\xrightarrow{\alpha}(\nu\,\tilde{n})Q''$. Therefore $((\nu\,\tilde{n})P'', (\nu\,\tilde{n})Q'')\in\mathcal{R}$;

2. α is an output action of the form $\alpha=(\nu\tilde{v})\bar{m}\langle\tilde{u}\rangle$ where $\bar{m}\notin\tilde{n}$ and $\tilde{v}_1=\tilde{n}\cap(\tilde{u}-\tilde{v})\neq\emptyset$, and $P\xrightarrow{\bar{m}\langle\tilde{u}\rangle}P''$. By $P\mathcal{S}Q$, we have $Q\xrightarrow{\bar{m}\langle\tilde{u}\rangle}Q''$ and $P''\mathcal{S}Q''$. Let $\tilde{v}_2=\tilde{n}-\tilde{v}_1$, by rule str-SCP3 and str-SUM2, $(\nu\,\tilde{n})P\equiv(\nu\,\tilde{v}_1)(\nu\,\tilde{v}_2)P$ and $(\nu\,\tilde{n})Q\equiv(\nu\,\tilde{v}_1)(\nu\,\tilde{v}_2)Q$. By the tr_OUT, we got $(\nu\,\tilde{v}_1)(\nu\,\tilde{v}_2)P\xrightarrow{\alpha}(\nu\,\tilde{v}_2)P''$ and $(\nu\,\tilde{v}_1)(\nu\,\tilde{v}_2)Q\xrightarrow{\alpha}(\nu\,\tilde{v}_2)Q''$, however $((\nu\,\tilde{v}_2)P'',(\nu\,\tilde{v}_2)Q'')\in\mathcal{R}$.

By the definition of οτ-bisimulation $\mathcal{S}$, we have $\mathcal{R}\subseteq\mathcal{S}$. ■

The οτ-bisimulation gives a measurement on processes' states by observing available reductions and output actions, but cannot determine how a process responds to incoming messages, since input actions are not observed. To determine responsive behaviours, we introduce a new term for specifying input messages.

> **Notation 3−14**: We add the auxiliary P-term $[\dagger m\langle\tilde{u}\rangle]P$, the *localisation* of the sent message $\bar{m}\langle\tilde{u}\rangle$ with process P, into the process syntax. Properties for this term are shown in Figure 3-2.

The term $[\dagger m\langle\tilde{u}\rangle]P$ is not for modelling processes, but only designed to express responsive bisimulation relations between processes. It couples process P with the message $\tilde{u}$ which is buffered in channel m and unobservable from outside, even though the output polar $\bar{m}$ may have been known by outsiders. The essential purpose of this term is to hide the message $\bar{m}\langle\tilde{u}\rangle$ from external observers, so that it will not be mistaken as an output from P. This will be discussed further in Session 5.

Structural equivalence :

lStr_NULL $[{}^{\dagger}m\langle\tilde{u}\rangle]\,\mathbf{0} \equiv \mathbf{0}$; **lStr_IND** $([{}^{\dagger}m\langle\tilde{u}\rangle]P) \mid Q \equiv [{}^{\dagger}m\langle\tilde{u}\rangle](P \mid Q)$, if $m \notin fn(Q)$

lStr_LOC $(\nu\, m)[{}^{\dagger}m\langle\tilde{u}\rangle]P \equiv (\nu\, m)(\bar{m}\langle\tilde{u}\rangle \mid P)$; **lStr_SUM2′** $[{}^{\dagger}m\langle\tilde{u}\rangle][{}^{\dagger}n\langle\tilde{v}\rangle]P \equiv [{}^{\dagger}n\langle\tilde{v}\rangle][{}^{\dagger}m\langle\tilde{u}\rangle]P$

Transition :

lTr_SYNC2 $\dfrac{P \xrightarrow{{}^{\dagger}m(\tilde{u})} P'}{[{}^{\dagger}m\langle\tilde{u}\rangle]P \xrightarrow{\tau} P'}$, **lTr_INV** $\dfrac{P \xrightarrow{\alpha} P' \qquad \alpha \neq {}^{\dagger}m(\tilde{u})}{[{}^{\dagger}m\langle\tilde{u}\rangle]P \xrightarrow{\alpha} [{}^{\dagger}m\langle\tilde{u}\rangle]P'}$

Figure 3-2 Localised output action.

The rule lTr_SYNC2 added a new case for defining the τ action. Unlike in rule tr_INTL, here the name restriction is not required. However, since only the input polar, ${}^{\dagger}m$, of the channel name m is involved, and the preservation of τ actions is maintained by input prefixing.

Corollary 3−15: The following conclusion can be immediately drew from the rules in Figure 3-2:
(1) If $P \xrightarrow{\bar{m}\langle\tilde{u}\rangle} P'$ then $(\nu\, m)P \equiv (\nu\, m)[{}^{\dagger}m\langle\tilde{u}\rangle]P'$; (3) $P{\downarrow}{}^{\dagger}m$ implies $([{}^{\dagger}m\langle\tilde{u}\rangle]P){\downarrow}\tau$;
(2) $P\not\downarrow\alpha$ implies $([{}^{\dagger}m\langle\tilde{u}\rangle]P)\not\downarrow\alpha$ if $\alpha\neq\tau$, or, $\alpha=\tau$ but $P\not\downarrow{}^{\dagger}m$; (4) $([{}^{\dagger}m\langle\tilde{u}\rangle]P)\not\downarrow{}^{\dagger}m(\tilde{u})$.

Now we can begin to introduce new behavioural equivalence relations.

Definition 3−16: Let $\mathcal{T}[.]$ be the **responsive testing context** of syntax $\mathcal{T} ::= [.] \mid [{}^{\dagger}m\langle\tilde{u}\rangle]\mathcal{T}$, then we define strong and weak **responsive equivalence**s as: $P \simeq_r Q$ iff $\forall\mathcal{T}.(\mathcal{T}[P] \sim_{o\tau} \mathcal{T}[Q])$; $P \approxeq_r Q$ iff $\forall\mathcal{T}.(\mathcal{T}[P] \approx_{o\tau} \mathcal{T}[Q])$.

This definition gives a quite clear description about the meaning of equivalence in responsive behaviour, but is not so useful since it requires the exhaustive testing over the infinite set of responsive testing contexts. A more practical definition is the r1-bisimulation, so named because it is structurally comparable to the 1-bisimulation in [Amadio96].

Definition 3−17: A strong (or weak) r1-**bisimulation** is a strong (or weak, respectively) oτ-bisimulation $\mathcal{S}$ such that whenever $P\mathcal{S}Q$ then $[{}^{\dagger}m\langle\tilde{u}\rangle]P\,\mathcal{S}\,[{}^{\dagger}m\langle\tilde{u}\rangle]Q$ for all $[{}^{\dagger}m\langle\tilde{u}\rangle]$.
We denote the largest strong r1-bisimulation as $\sim_{r1}$, and the largest weak r1-bisimulation as $\approx_{r1}$.

Lemma 3−18: Responsive equivalence and r1-bisimulation coincide, that is, $\sim_{r1} \equiv \simeq_r$ and $\approx_{r1} \equiv \approxeq_r$.
Proof: We only show it for the strong case here, and the weak case can be proven in similar way.

$\sim_{r1} \subseteq \simeq_r$: Proven by induction. Let $P \sim_{r1} Q$, then we can write $\mathcal{T}_0[P] \sim_{r1} \mathcal{T}_0[Q]$ where $\mathcal{T}_0 \stackrel{\text{def}}{=} [.]$.
Assume $\mathcal{T}_i[P] \sim_{r1} \mathcal{T}_i[Q]$ is held for some responsive testing context $\mathcal{T}_i$.
By the definition of $\sim_{r1}$, for all $[{}^{\dagger}m\langle\tilde{u}\rangle]$, we have $\mathcal{T}_{i1}[P] \sim_{r1} \mathcal{T}_{i1}[Q]$ for each $\mathcal{T}_{i1} \stackrel{\text{def}}{=} [{}^{\dagger}m\langle\tilde{u}\rangle]\mathcal{T}_i$.
Since $\sim_{r1} \subseteq \sim_{o\tau}$ from the definition of $\sim_{r1}$, we can conclude that, $P \sim_{r1} Q$ implies $\forall\mathcal{T}.(\mathcal{T}[P] \sim_{o\tau} \mathcal{T}[Q])$, that is, $P \simeq_r Q$, by the definition of $\simeq_r$.

$\sim_{r1} \supseteq \simeq_r$: Let $P \simeq_r Q$, then $P \sim_{o\tau} Q$, because $\mathcal{T}_0[P] \sim_{o\tau} \mathcal{T}_0[Q]$ for $\mathcal{T}_0 \stackrel{\text{def}}{=} [.]$.
Also, we have $[{}^{\dagger}m\langle\tilde{u}\rangle]P \sim_{o\tau} [{}^{\dagger}m\langle\tilde{u}\rangle]Q$ for all $[{}^{\dagger}m\langle\tilde{u}\rangle]$, because $\mathcal{T}_1[P] \sim_{o\tau} \mathcal{T}_1[Q]$ for each $\mathcal{T}_1 \stackrel{\text{def}}{=} [{}^{\dagger}m\langle\tilde{u}\rangle][.]$.
This implies $\forall\mathcal{T}.(\mathcal{T}[[{}^{\dagger}m\langle\tilde{u}\rangle]P] \sim_{o\tau} \mathcal{T}[[{}^{\dagger}m\langle\tilde{u}\rangle]Q])$, since $\mathcal{T}_2[P] \sim_{o\tau} \mathcal{T}_2[Q]$ for each $\mathcal{T}_2 \stackrel{\text{def}}{=} \mathcal{T}[\mathcal{T}_1[.]]$. I.e., $P \simeq_r Q$ implies $[{}^{\dagger}m\langle\tilde{u}\rangle]P \simeq_r [{}^{\dagger}m\langle\tilde{u}\rangle]Q$ for all $[{}^{\dagger}m\langle\tilde{u}\rangle]$. Therefore $\simeq_r \subseteq \sim_{r1}$ by the definition of $\sim_{r1}$. ■

It is easy to verify that $O_1 \approx_{r1} O_2$ holds for the previously mentioned examples. The r1-bisimulation provides a test platform and measures behavioural equivalence from outside of target processes.

While responsive equivalence and r1-bisimulation provide a good base for describing similarities of responsive behaviours, it can tell little about why or when two processes may offer similar behaviours. For closer study, we need an inside view observing input actions.

Definition 3−19: A (strong) **responsive bisimulation** is a (strong) oτ-bisimulation $\mathcal{S}$ such that whenever $P\mathcal{S}Q$ then $P\xrightarrow{\dagger m(\tilde{u})}P'$ implies either $Q\xrightarrow{\dagger m(\tilde{u})}Q'$ and $P'\mathcal{S}Q'$, or $Q\xrightarrow{\tau}Q'$ and $P'\mathcal{S}[\dagger m\langle\tilde{u}\rangle]Q'$. The weak version is obtained by replacing transitions with weak transitions everywhere. We denote $\sim_r$ and $\approx_r$ to be the largest strong and weak responsive bisimulation respectively. Clearly, $\sim_r\subseteq\approx_r$.

For the previously mentioned example, we can also easily verify that $O_1\approx_r O_2$. It is no surprise, since:

Lemma 3−20: The responsive bisimulation and r1-bisimulation coincide, that is, $\sim_r\equiv\sim_{r1}$ and $\approx_r\equiv\approx_{r1}$.
Proof: We only show that for the strong case here, and the weak case can be proven in a similar way.

$\sim_r\subseteq\sim_{r1}$: Let $\mathcal{R}\overset{\text{def}}{=}\{([\dagger m\langle\tilde{u}\rangle]P,[\dagger m\langle\tilde{u}\rangle]Q):P\mathcal{S}Q\}\cup\mathcal{S}$ for $\mathcal{S}=\sim_r$. Assume $[\dagger m\langle\tilde{u}\rangle]P\xrightarrow{\alpha}P'$ for some an arbitrary action α, by the rules in Figure 3-2 and by Corollary 3-15 (4), it is only possible in the following two cases:

(1) $P\xrightarrow{\alpha}P''$ and $\alpha\neq\dagger m(\tilde{u})$, then $P'=[\dagger m\langle\tilde{u}\rangle]P''$. Since $P\sim_r Q$, we have
either $Q\xrightarrow{\alpha}Q''$, $P''\sim_r Q''$ and $[\dagger m\langle\tilde{u}\rangle]Q\xrightarrow{\alpha}[\dagger m\langle\tilde{u}\rangle]Q''$, and therefore $([\dagger m\langle\tilde{u}\rangle]P'',[\dagger m\langle\tilde{u}\rangle]Q'')\in\mathcal{R}$;
or $\alpha=\dagger n(\tilde{v})$, $Q\xrightarrow{\tau}Q''$, $P''\sim_r[\dagger n\langle\tilde{v}\rangle]Q''$, that is, $[\dagger m\langle\tilde{u}\rangle]P\xrightarrow{\dagger n(\tilde{v})}[\dagger m\langle\tilde{u}\rangle]P''$ and $[\dagger m\langle\tilde{u}\rangle]Q\xrightarrow{\tau}[\dagger m\langle\tilde{u}\rangle]Q''$. By rule lStr_SUM2′, we have $[\dagger n\langle\tilde{v}\rangle][\dagger m\langle\tilde{u}\rangle]Q''\equiv[\dagger m\langle\tilde{u}\rangle][\dagger n\langle\tilde{v}\rangle]Q''$, therefore $([\dagger m\langle\tilde{u}\rangle]P'',[\dagger n\langle\tilde{v}\rangle][\dagger m\langle\tilde{u}\rangle]Q'')\in\mathcal{R}$;

(2) $\alpha=\tau$ and $P\xrightarrow{\dagger m(\tilde{u})}P'$, then by $P\sim_r Q$ it implies
either $Q\xrightarrow{\dagger m(\tilde{u})}Q'$, $[\dagger m\langle\tilde{u}\rangle]Q\xrightarrow{\tau}Q'$ and $P'\sim_r Q'$, that is $(P',Q')\in\mathcal{R}$ since $\mathcal{R}\supseteq\sim_r$;
or $Q\not\xrightarrow{\dagger m(\tilde{u})}$, then $Q\xrightarrow{\tau}Q'$ and $P'\sim_r[\dagger m\langle\tilde{u}\rangle]Q'$, that is $(P',[\dagger m\langle\tilde{u}\rangle]Q')\in\mathcal{R}$ since $\mathcal{R}\supseteq\sim_r$;

Then by definition of $\sim_r$, we have $\mathcal{R}\subseteq\sim_r$, that is, $P\sim_r Q$ implies $[\dagger m\langle\tilde{u}\rangle]P\sim_r[\dagger m\langle\tilde{u}\rangle]Q$. Because $\sim_r\subseteq\sim_{o\tau}$, and because $[\dagger m\langle\tilde{u}\rangle]$ is arbitrary here, we have $\sim_r\subseteq\sim_{r1}$ by the definition of $\sim_{r1}$.

$\sim_r\supseteq\sim_{r1}$: Let $P\sim_{r1}Q$, then $[\dagger m\langle\tilde{u}\rangle]P\sim_{r1}[\dagger m\langle\tilde{u}\rangle]Q$ for all $[\dagger m\langle\tilde{u}\rangle]$. Assume $P\xrightarrow{\alpha}P'$ for some action α:

If α is a non-input action, then $Q\xrightarrow{\alpha}Q'$ and $P'\sim_{r1}Q'$;

If α is an input action, $\alpha=\dagger m(\tilde{u})$, then $[\dagger m\langle\tilde{u}\rangle]P\xrightarrow{\tau}P'$. By $[\dagger m\langle\tilde{u}\rangle]P\sim_{r1}[\dagger m\langle\tilde{u}\rangle]Q$ and $[\dagger m\langle\tilde{u}\rangle]Q\xrightarrow{\tau}Q''$, it must be
either $Q\xrightarrow{\dagger m(\tilde{u})}Q'$ and $Q''\equiv Q'$. But $P'\sim_{o\tau}Q'$ since $\sim_{r1}\subseteq\sim_{o\tau}$. Let $\mathcal{R}_1=\{(P,Q),(P',Q')\}$ then $\mathcal{R}_1\subseteq\sim_{o\tau}$;
or $Q\xrightarrow{\tau}Q'$ and $Q''\equiv[\dagger m\langle\tilde{u}\rangle]Q'$. But $P'\sim_{o\tau}[\dagger m\langle\tilde{u}\rangle]Q'$ since $\sim_{r1}\subseteq\sim_{o\tau}$. Let $\mathcal{R}_2=\{(P,Q),(P',[\dagger m\langle\tilde{u}\rangle]Q')\}$, then $\mathcal{R}_2\subseteq\sim_{o\tau}$;

Let $\mathcal{R}=\sim_{r1}\cup\mathcal{R}_1\cup\mathcal{R}_2$, then $\mathcal{R}\subseteq\sim_{o\tau}$ since $\sim_{r1}\subseteq\sim_{o\tau}$, then we have $\mathcal{R}\subseteq\sim_r$ by the definition of $\sim_r$. ■

Corollary 3−21: The responsive bisimulation and responsive equivalence coincide: $\sim_r\equiv\simeq_r$, $\approx_r\equiv\approxeq_r$.

That is, the two different viewpoints mentioned in the introduction, are united into one concept.

Another interesting coclusion is:

Proposition 3−22: $P\approx_r(\nu n)(!\dagger m(\tilde{x}).\bar{n}\langle\tilde{x}\rangle\mid P\{{}^{n}/_{m}\})$ for all P and m, where n is a fresh name for P.
Proof: It is trivial to varify. ■

4 Properties of the responsive bisimulation

We now investigate some formal properties of responsive bisimulation.

Corollary 4−23: The responsive bisimulations are preserved by localisation. That is, let $\mathcal{S}$ be either $\sim_r$ or $\approx_r$, then $P\mathcal{S}Q$ implies $[\dagger m\langle\tilde{u}\rangle]P\,\mathcal{S}\,[\dagger m\langle\tilde{u}\rangle]Q$ for all $[\dagger m\langle\tilde{u}\rangle]$.

Proof: Let $\mathcal{R}$ be either $\sim_{r1}$ or $\approx_{r1}$, corresponding to $\mathcal{S}$ respectively, then by Lemma 3-20, $P\mathcal{S}Q$ implies $P\mathcal{R}Q$, which then implies that $[\dagger m\langle\tilde{u}\rangle]P\,\mathcal{R}\,[\dagger m\langle\tilde{u}\rangle]Q$ for all $[\dagger m\langle\tilde{u}\rangle]$ according to the definition of r1-bisimulation, then again by Lemma 3-20, we have $[\dagger m\langle\tilde{u}\rangle]P\,\mathcal{S}\,[\dagger m\langle\tilde{u}\rangle]Q$. ■

Lemma 4−24: The responsive bisimulations are equivalences.

Proof: Here we only give the proof for $\sim_r$, and the weak case can be proven similarly.

Reflexive : $P\sim_r P$ for any P, according to the definition of $\sim_r$;

Symmetric : if $P\sim_r Q$ then $Q\sim_r P$, by the definition of $\sim_r$;

Transitive : Let $P_1\mathcal{R}_1P_2$ and $P_2\mathcal{R}_2P_3$, where $\mathcal{R}_1\subseteq\sim_r$ and $\mathcal{R}_2\subseteq\sim_r$, and therefore $P_1(\mathcal{R}_1\mathcal{R}_2)P_3$. For arbitrary action α, such that $P_1\xrightarrow{\alpha}P_1'$,

If α is not an input communication act, then

$P_1\xrightarrow{\alpha}P_1'$ implies $P_2\xrightarrow{\alpha}P_2'$ and $P_1'\mathcal{R}_1P_2'$, which then implies $P_3\xrightarrow{\alpha}P_3'$ and $P_2'\mathcal{R}_2P_3'$, ie., $P_1'(\mathcal{R}_1\mathcal{R}_2)P_3'$.

If α is an input communication act, say $\alpha=\dagger m(\tilde{u})$, and $P_1\xrightarrow{\dagger m(\tilde{u})}P_1'$, then we may have either

$P_2\xrightarrow{\dagger m(\tilde{u})}P_2'$ and $P_1'\mathcal{R}_1P_2'$, $P_3\xrightarrow{\dagger m(\tilde{u})}P_3'$ and $P_2'\mathcal{R}_2P_3'$, and therefore $P_1'(\mathcal{R}_1\mathcal{R}_2)P_3'$;

or $P_2\xrightarrow{\dagger m(\tilde{u})}P_2'$ and $P_1'\mathcal{R}_1P_2'$, $P_3\xrightarrow{\tau}P_3'$ and $P_2'\mathcal{R}_2[\dagger m\langle\tilde{u}\rangle]P_3'$, and therefore $P_1'(\mathcal{R}_1\mathcal{R}_2)[\dagger m\langle\tilde{u}\rangle]P_3'$;

or $P_2\xrightarrow{\tau}P_2'$ and $P_1'\mathcal{R}_1[\dagger m\langle\tilde{u}\rangle]P_2'$, $P_3\xrightarrow{\tau}P_3'$ and $P_2'\mathcal{R}_2P_3'$. By $\sim_r\equiv\sim_{r1}$, we have $\mathcal{R}_2\subseteq\sim_{r1}$, so $[\dagger m\langle\tilde{u}\rangle]P_2'\mathcal{R}_2[\dagger m\langle\tilde{u}\rangle]P_3'$, and therefore $P_1'(\mathcal{R}_1\mathcal{R}_2)[\dagger m\langle\tilde{u}\rangle]P_3'$. By the definition, $(\mathcal{R}_1\mathcal{R}_2)\subseteq\sim_r$. ■

A problem is apparent: the responsive bisimulation is not preserved by parallel composition in general. For instance, with O_1 and O_2 of the earlier example, we have $O_1\approx_r O_2$, but $(O_1|O_3)\not\approx_r(O_2|O_3)$ for $O_3\stackrel{\text{def}}{=}!\dagger m(\tilde{x}).R$, because the occurrence of input polar $\dagger m$ in O_3 has changed the ability of O_1 to receive messages on $\dagger m$. However, as mentioned at the beginning of this paper, the purpose of our study is about object modelling, and as the nature of object systems, the ownership of each input port should be unique. For example, the identity of an object is uniquely owned by no one else but that object; each method of each object is also uniquely identified so that no message would be delivered to wrong destination. In general, each input polar has a restricted scope (or ownership), and is never exported outside this scope.

When responsive bisimulation is strictly restricted within objects modelling, the problem domain where it is needed, then its preservation in parallel composition can be guaranteed. To show this, we first formalise the restriction needed on input polars.

Definition 4−25: Let $\dagger m$ be the input polar of a communication channel name m, P be a process for which $m\in fin(P)$, and $\mathcal{E}$ be the context $\mathcal{E}[.]\stackrel{\text{def}}{=}(\nu\tilde{n})\,(Env\,|\,[.]\,)$ where $\dagger m\notin fin(Env)$ while m may or may not be a member of $\tilde{n}$. We say that,

P is an *owner* of $\dagger m$ (or say, $\dagger m$ is owned by P) with respect to the *environment Env*;

Env is an environment free of $\dagger m$ (or say, $\dagger m$*-free environment*);

$\mathcal{E}[.]$ is an $\dagger m$*-safe* environment context, or $\dagger m$-safe environment for short.

An $\dagger m$-safe environment only allows the process in the hole to consume a message sent along the channel m, ensuring no interference from the environment. It reflects the fact that the responsive behaviour of a process can be measured only when messages sent to it are guaranteed not to be intercepted by some other process.

Definition 4−26: A process P is *safe* for Env, and the environment Env is said to be *safe* for P, if P is the owner of all ${}^{\dagger}m \in \mathit{fin}(P)$ with respect to the environment Env, that is, $\mathit{fin}(P) \cap \mathit{fin}(Env) = \emptyset$. We may call P a *safe process*, when the behaviour of P is only considered within environments which are safe for P.

A process P is *autonomous* if $\mathit{fin}(P) = \emptyset$.

Lemma 4−27: Evolution preserves process safety. I.e., if $\mathit{fin}(P) \cap \mathit{fin}(Env) = \emptyset$ holds for some process P and Env, then $\mathit{fin}(P') \cap \mathit{fin}(Env') = \emptyset$ also holds for all P' and Env', which are derivatives of P and Env respectively.

Proof: Simply because the input polar of a channel cannot be transmitted by communication. ■

Corollary 4−28: An autonomous process and all its derivatives are safe for any system.

When modelling objects in the π_p-calculus, all method bodies can be considered as autonomous, since after parameters passed through the method interface, further input (if any) can only be performed via channels that were initially private and informed to the senders by the forked method body. An object process itself is initially autonomous until creation, when its name (the unique identification) is exported to its environment. Its method names can also be considered as initially private to the object, and then exported to the caller during each method call. For example, similar to [Walker95] and [Zhang97] amongst others, the method call $\mathtt{o.m_1(a_1,a_2)}$ may be modelled as $(\nu\, mset)(\bar{o}\langle {}^{\dagger}mset\rangle \,|\, {}^{\dagger}mset(\tilde{m}).\bar{m}_1\langle \bar{a}_1,\bar{a}_2\rangle)$, and on the object side the encoding will look like $(\nu\, \tilde{m})(!{}^{\dagger}o({}^{\dagger}mset).\overline{mset}\langle\tilde{m}\rangle \,|\, \prod !{}^{\dagger}m_i(\tilde{x}).Body_i)$, where method names $\tilde{m}$ are initially private.

Proposition 4−29: The responsive bisimulations are preserved by parallel composition for safe processes. That is, let $\mathcal{S}$ be either $\sim_r$ or $\approx_r$, then $P_1 \mathcal{S} P_2$ implies $P_1|P \,\mathcal{S}\, P_2|P$ for all P which satisfy $(\mathit{fin}(P_1) \cup \mathit{fin}(P_2)) \cap \mathit{fin}(P) = \emptyset$.

Proof: Here we only show that for $\approx_r$. The strong case can be proven similarly.

Let $\equiv_P$ be the congruence induced by the commutativity and associativity laws for parallel composition "$|$" in Figure 2-1 and rule lStr_SUM2′ in Figure 3-2, and let relation $\mathcal{R} \overset{\text{def}}{=} \{(P_1|P, P_2|P) : (P_1 \approx_r P_2) \wedge (\mathit{fin}(P) \cap (\mathit{fin}(P_1|P_2)) = \emptyset)\} \cup \approx_r$. Let $Q_1 \overset{\text{def}}{=} P_1|P$ and $Q_2 \overset{\text{def}}{=} P_2|P$, and $Q_1 \overset{\alpha}{\Longrightarrow} Q_1'$ for some action α, by $P_1 \approx_r P_2$, it must in one of the following three cases:

(1) $P_1 \overset{\alpha}{\Longrightarrow} P_1'$, $P_2 \overset{\alpha}{\Longrightarrow} P_2'$ and $P_1' \approx_r P_2'$, and therefore $Q_1' \equiv P_1' \,|\, P$, $Q_2 \overset{\alpha}{\Longrightarrow} Q_2'$ for $Q_2' \overset{\text{def}}{=} P_2' \,|\, P$;
(2) $P \overset{\alpha}{\Longrightarrow} P'$, and therefore $Q_1' \equiv P_1 \,|\, P'$ and $Q_2 \overset{\alpha}{\Longrightarrow} Q_2'$ for $Q_2' \overset{\text{def}}{=} P_2 \,|\, P'$;
(3) $\alpha = {}^{\dagger}m(\tilde{u})$, $P_1 \overset{{}^{\dagger}m(\tilde{u})}{\Longrightarrow} P_1'$, $P_2 \overset{\tau}{\Longrightarrow} P_2'$ and $P_1' \approx_r [{}^{\dagger}m\langle\tilde{u}\rangle]P_2'$, by the safety condition $\mathit{fin}(P) \cap (\mathit{fin}(P_1) \cup \mathit{fin}(P_2)) = \emptyset$, it must be $m \notin \mathit{fin}(P)$ and therefore $[{}^{\dagger}m\langle\tilde{u}\rangle]P_2' \,|\, P \equiv [{}^{\dagger}m\langle\tilde{u}\rangle](P_2' \,|\, P)$, that is, $Q_1' \equiv P_1' \,|\, P$, and $Q_2 \overset{\tau}{\Longrightarrow} Q_2'$ for $Q_2' \overset{\text{def}}{=} P_2' \,|\, P$.

The cases where either $P_1 \overset{(\nu\tilde{v})\bar{m}\langle\tilde{u}\rangle}{\Longrightarrow} P_1'$ and $P \overset{{}^{\dagger}m(\tilde{u})}{\Longrightarrow} P'$, or $P_1 \overset{{}^{\dagger}m(\tilde{u})}{\Longrightarrow} P_1'$ and $P \overset{(\nu\tilde{v})\bar{m}\langle\tilde{u}\rangle}{\Longrightarrow} P'$, have been covered by theses three above cases, according to Remark 2-3, and therefore need not to be considered separately.

Since $(Q_1', Q_2') \in \mathcal{R}$ for cases 1-2, and $(Q_1', [{}^{\dagger}m\langle\tilde{u}\rangle]Q_2') \in \mathcal{R} \circ \equiv_P$ for the third case, therefore $\mathcal{R}$ is a $\approx_r$ upto $\equiv_P$. ■

Let σ denote a name substitution which is over output polars only, otherwise standard. Whenever applied to a process or an action, bound names (in pairs of both polars) are automatically renamed to avoid conflict. We do not need to consider substitution over input polars because they can not be sent through channels in the π_p-calculus. Clearly the safety of processes is preserved by the output polar substitution.

Proposition 4−30: The responsive bisimulations are preserved by output polar substitution. That is, let $\mathcal{S}$ be either $\sim_r$ or $\approx_r$, then for all $\sigma = \{\tilde{u}/\tilde{x}\}$, $P \mathcal{S} Q$ implies $P\sigma \mathcal{S} Q\sigma$.

Proof: These can be proven by showing $\mathcal{R} \overset{\text{def}}{=} \{(P\sigma, Q\sigma) : P \mathcal{S} Q\} \cup \mathcal{S}$ is a $\mathcal{S}$ upto $\equiv$, where $\equiv$ be the structural congruence in Figure 2-1 and Figure 3-2. Here we only show that for $\mathcal{S} \subseteq \sim_r$, and the weak case can be proven similarly. Lets exam all the possible actions that $P\sigma$ may take:

$P\sigma \xrightarrow{(\nu\tilde{s})\overline{m}\langle\tilde{u}\rangle} P''$: it is only possible when $P\sigma \equiv (\nu\tilde{s})(\overline{m}\langle\tilde{u}\rangle \mid P'')$, by the transition rules listed in Figure 2-2 and Figure 3-2. Remember the implicity renaming over fresh names to avoid name clash, and notice that the substitution only effects to free output polars, then there must exist some $\overline{n}$, $\tilde{v}$ and P' such that $\overline{m} = \overline{n}\sigma$, $\tilde{u} = \tilde{v}\sigma$, $P'' = P'\sigma$ and $P \equiv (\nu\tilde{s})(\overline{n}\langle\tilde{v}\rangle \mid P')$. Clearly, $P \xrightarrow{(\nu\tilde{s})\overline{n}\langle\tilde{v}\rangle} P'$, it implies $Q \xrightarrow{(\nu\tilde{s})\overline{n}\langle\tilde{v}\rangle} Q'$ and $P' \sim_r Q'$ since $P \sim_r Q$, this further implies there exists some Q' s.t. $Q \equiv (\nu\tilde{s})(\overline{n}\langle\tilde{v}\rangle \mid Q')$. Therefore, $Q\sigma \equiv (\nu\tilde{s})(\overline{m}\langle\tilde{u}\rangle \mid Q'\sigma)$ and $Q\sigma \xrightarrow{(\nu\tilde{s})\overline{m}\langle\tilde{u}\rangle} Q'\sigma$. This matches $(P'\sigma, Q'\sigma) \in \mathcal{R}$ as required.

$P\sigma \xrightarrow{{}^{\dagger}m(\tilde{u})} P''$: it is only possible when $P \downarrow {}^{\dagger}m(\tilde{v})$ where $\tilde{v}$ satisfies $\tilde{u} = \tilde{v}\sigma$. Let $P \xrightarrow{{}^{\dagger}m(\tilde{v})} P'$, it is easy to verify that $P'' \equiv P'\sigma$. Since P must be of the form either $P \equiv (\nu\tilde{s})(\,!{}^{\dagger}m(\tilde{x}).P_1 \mid P_2)$ or $P \equiv (\nu\tilde{s})((\,{}^{\dagger}m(\tilde{x}).P_1 + G) \mid P_2)$, then by $P \sim_r Q$, this implies

either $Q \xrightarrow{{}^{\dagger}m(\tilde{v})} Q'$ and $P' \sim_r Q'$, then we have $(P'\sigma, Q'\sigma) \in \mathcal{R}$ as required, and it is easy to verify that $Q\sigma \xrightarrow{{}^{\dagger}m(\tilde{u})} Q'\sigma$, with the same way as above;

or $Q \xrightarrow{\tau} Q'$ and $P' \sim_r [{}^{\dagger}m\langle\tilde{v}\rangle]Q'$. Since in the polar π-calculus, as in normal π-calculus, the channel name of an internal action τ is always bound, so $Q\sigma \xrightarrow{\tau} Q'\sigma$. Notice $([{}^{\dagger}m\langle\tilde{v}\rangle]Q')\sigma \equiv [{}^{\dagger}m\langle\tilde{u}\rangle]Q'\sigma$, we have $(P'\sigma, ([{}^{\dagger}m\langle\tilde{v}\rangle]Q')\sigma) \in \mathcal{R}$, or, $(P'\sigma, [{}^{\dagger}m\langle\tilde{u}\rangle](Q'\sigma)) \in \mathcal{R}$;

$P\sigma \xrightarrow{\tau} P''$: it is possible only when there exist some process P_1 and complementary output-input action pair $(\nu\tilde{v})\overline{m}\langle\tilde{u}\rangle$ and ${}^{\dagger}m(\tilde{u})$ such that, $P\sigma \equiv (\nu\tilde{v}, m)(P_1 \mid \overline{m}\langle\tilde{u}\rangle)$, $P_1 \xrightarrow{{}^{\dagger}m(\tilde{u})} P_1''$, and $P'' \equiv (\nu\tilde{v}, m)P_1''$.

Clearly, there must exist some P_2 and $\tilde{s}$ such that $P_1 = P_2\sigma$, $\tilde{u} = \tilde{s}\sigma$, and $P \equiv (\nu\tilde{v}, m)(P_2 \mid \overline{m}\langle\tilde{s}\rangle)$. That is, $\overline{m}\langle\tilde{s}\rangle \xrightarrow{\overline{m}\langle\tilde{s}\rangle} \mathbf{0}$, $P_2 \xrightarrow{{}^{\dagger}m(\tilde{s})} P_2'$, then, by rule tr_INTL, we have $P \xrightarrow{\tau} P'$, where $P' \equiv (\nu\tilde{v}, a)P_2'$.

By $P \sim_r Q$, this implies $Q \xrightarrow{\tau} Q'$ and $P' \sim_r Q'$. This is only possible when there exists some Q_2 and complementary output-input action pair $(\nu\tilde{w})\overline{n}\langle\tilde{r}\rangle$ and ${}^{\dagger}n(\tilde{r})$ such that, $Q \equiv (\nu\tilde{w}, n)(Q_2 \mid \overline{n}\langle\tilde{r}\rangle)$, where $Q_2 \xrightarrow{{}^{\dagger}n(\tilde{r})} Q_2'$, therefore $Q\sigma \equiv (\nu\tilde{w}, n)(Q_2\sigma \mid \overline{n}\langle\tilde{r}\sigma\rangle)$ and $Q_2 \xrightarrow{{}^{\dagger}n(\tilde{r}\sigma)} Q_2'\sigma$. By rule tr_INTL, $Q\sigma \xrightarrow{\tau} Q'\sigma$, and again we have $(P'\sigma, Q'\sigma) \in \mathcal{R}$ as required. ■

Corollary 4−31: The responsive bisimulations are preserved by input prefix. That is, let $\mathcal{S}$ be either $\sim_r$ or $\approx_r$, then $P \mathcal{S} Q$ implies ${}^{\dagger}m(\tilde{u}).P \,\mathcal{S}\, {}^{\dagger}m(\tilde{u}).Q$ for all ${}^{\dagger}m(\tilde{u})$.

Proofs for other properties of the responsive bisimulations also have the similar difference with those for standard bisimulations, and we have to provide them instantly.

Proposition 4−32: The responsive bisimulations are preserved by restriction. That is, let $\mathcal{S}$ be either $\sim_r$ or $\approx_r$, then $P \mathcal{S} Q$ implies $(\nu\tilde{v})P \,\mathcal{S}\, (\nu\tilde{v})Q$ for all $\tilde{v}$.

Proof: These can be proven by showing $\mathcal{R} \stackrel{\text{def}}{=} \{((\nu\tilde{v})P, (\nu\tilde{v})Q) : P \mathcal{S} Q\} \cup \mathcal{S}$ is a $\mathcal{S}$ upto ≡, where ≡ be the structural congruence in Figure 2-1 and Figure 3-2. Here we only show that for strong case $\mathcal{S} \subseteq \sim_r$, and the weak case can be proven similarly. Lets exam all the possible transitions that $(\nu\tilde{v})P$ may take:

$(\nu\tilde{v})P \xrightarrow{(\nu\tilde{s})\overline{m}\langle\tilde{u}\rangle} P''$: Giving the implicity renaming has removed all fresh name clash, let $\tilde{v}_1 = \tilde{v} \cap \tilde{u}$ and $\tilde{v}_2 = \tilde{v} - \tilde{v}_1$, then we have $(\nu\tilde{v})P \equiv (\nu\tilde{v}_1)(\nu\tilde{v}_2)P$. From the structural congruence rules and transition rules, this transition is only possible when $m \notin \tilde{v}$, $\tilde{v}_1 \subseteq \tilde{s}$ and there exists some P' such that $P \equiv (\nu\tilde{r})(P' \mid \overline{m}\langle\tilde{u}\rangle)$ where $\tilde{r} = \tilde{s} - \tilde{v}_1$. By rule tr_OUT, tr_RES and tr_PARL, we have $P \xrightarrow{(\nu\tilde{r})\overline{m}\langle\tilde{u}\rangle} P'$ and $P'' \equiv (\nu\tilde{v}_2)\, P'$. This implies $Q \xrightarrow{(\nu\tilde{r})\overline{m}\langle\tilde{u}\rangle} Q'$ and $P' \sim_r Q'$ since $P \sim_r Q$, by rule tr_OUT and tr_RES we have $(\nu\tilde{v})Q \xrightarrow{(\nu\tilde{s})\overline{m}\langle\tilde{u}\rangle} (\nu\tilde{v}_2)Q'$, it matches $((\nu\tilde{v}_2)P', (\nu\tilde{v}_2)Q') \in \mathcal{R}$ as required.

$(\nu\tilde{v})P \xrightarrow{{}^{\dagger}m(\tilde{u})} P''$: it is only possible when $m \notin \tilde{v}$ and $P \downarrow {}^{\dagger}m(\tilde{u})$. Let $P \xrightarrow{{}^{\dagger}m(\tilde{u})} P'$, by rule tr_RES, $(\nu\tilde{v})P \xrightarrow{{}^{\dagger}m(\tilde{u})} (\nu\tilde{v})P'$, that is, $P'' \equiv (\nu\tilde{v})P'$. From $P \sim_r Q$, this implies either $Q \xrightarrow{{}^{\dagger}m(\tilde{u})} Q'$ and $P' \sim_r Q'$, then by rule tr_RES, $(\nu\,\tilde{v})Q \xrightarrow{{}^{\dagger}m(\tilde{u})} (\nu\,\tilde{v})Q'$, and $((\nu\,\tilde{v})P', (\nu\,\tilde{v})Q') \in \mathcal{R}$ as required;

or $Q \xrightarrow{\tau} Q'$ and $P' \sim_r [{}^{\dagger}m\langle\tilde{u}\rangle]Q'$, then $(\nu\,\tilde{v})Q \xrightarrow{\tau} (\nu\,\tilde{v})Q'$ from rule tr_RES, and we have $((\nu\tilde{v})P', (\nu\tilde{v})([{}^{\dagger}m\langle\tilde{u}\rangle]Q')) \in \mathcal{R}$ as required.

$(\nu\,\tilde{v})P \xrightarrow{\tau} P''$: it is only possible when $P \xrightarrow{\tau} P'$, then by rule tr_RES, $(\nu\,\tilde{v})P \xrightarrow{\tau} (\nu\,\tilde{v})P'$, that is, $P'' \equiv (\nu\,\tilde{v})P'$. By $P \sim_r Q$, we have $Q \xrightarrow{\tau} Q'$, and by rule tr_RES, $(\nu\,\tilde{v})Q \xrightarrow{\tau} (\nu\,\tilde{v})Q'$, and $((\nu\,\tilde{v})P', (\nu\,\tilde{v})Q') \in \mathcal{R}$ as required.

Put them together, by the definition, $\mathcal{R}$ is a $\sim_r$ upto $\equiv$. ■

Proposition 4−33: The responsive bisimulations are preserved by choice. That is, let $\mathcal{S}$ be either $\sim_r$ or $\approx_r$, then $G_1 \mathcal{S} G_2$ implies $(G_1+G)\mathcal{S}(G_2+G)$ for all G.

Proof: The proof is trivial, since for both sides the first action must be an input action in any case. ■

Proposition 4−34: The responsive bisimulations are preserved by replication for autonomous processes. That is, let $\mathcal{S}$ be either $\sim_r$ or $\approx_r$, P_1 and P_2 be autonomous processes, then $P_1 \mathcal{S} P_2$ implies $!{}^{\dagger}n(\tilde{x}).P_1 \mathcal{S}\, !{}^{\dagger}n(\tilde{x}).P_2$ for all input prefix ${}^{\dagger}n(\tilde{x})$, and $!\tau.P_1 \mathcal{S}\, !\tau.P_2$.

Proof: Let $\mathcal{A}_\mathrm{p}$ be the set of all safe processes in the environment concerned, $\mathcal{U}_\mathrm{ap}$ be the set of all autonomous processes, first we show that both

$$\mathcal{R}_1 \stackrel{\mathrm{def}}{=} \{(R_1 \,|\, !{}^{\dagger}n(\tilde{x}).P_1, R_2 \,|\, !{}^{\dagger}n(\tilde{x}).P_2) : (P_1,P_2 \in \mathcal{U}_\mathrm{ap}) \wedge (P_1 \mathcal{S} P_2) \wedge (R_1,R_2 \in \mathcal{A}_\mathrm{p}) \wedge (R_1 \mathcal{S} R_2)\} \cup \mathcal{S} \quad \text{and}$$

$$\mathcal{R}_2 \stackrel{\mathrm{def}}{=} \{(R_1 \,|\, !\tau.P_1, R_2 \,|\, !\tau.P_2) : (P_1,P_2 \in \mathcal{U}_\mathrm{ap}) \wedge (P_1 \mathcal{S} P_2) \wedge (R_1,R_2 \in \mathcal{A}_\mathrm{p}) \wedge (R_1 \mathcal{S} R_2)\} \cup \mathcal{S}$$

are $\mathcal{S}$ upto $\equiv$, then this proposition can be concluded by restricting $R_1 \stackrel{\mathrm{def}}{=} \mathbf{0}$ and $R_2 \stackrel{\mathrm{def}}{=} \mathbf{0}$. Here we only show these for strong case $\mathcal{S} \subseteq \sim_r$, and the weak case can be proven similarly.

For $\mathcal{R}_1$, we write $Q_1 \stackrel{\mathrm{def}}{=} R_1 \,|\, !{}^{\dagger}n(\tilde{x}).P_1$ and $Q_2 \stackrel{\mathrm{def}}{=} R_2 \,|\, !{}^{\dagger}n(\tilde{x}).P_2$, and exam all the possible transitions Q_1 may take:

$Q_1 \xrightarrow{{}^{\dagger}n(\tilde{u})} Q_1'$: since $R_1,R_2 \in \mathcal{A}_\mathrm{p}$ (this implies $n \notin fin(R_1 \,|\, R_2)$), it must be $Q_1' \equiv R_1 \,|\, P_1\{\tilde{u}/\tilde{x}\} \,|\, !{}^{\dagger}n(\tilde{x}).P_1$, and we can have $Q_2 \xrightarrow{{}^{\dagger}n(\tilde{u})} Q_2'$ where $Q_2' \equiv R_2 \,|\, P_2\{\tilde{u}/\tilde{x}\} \,|\, !{}^{\dagger}n(\tilde{x}).P_2$. Write $R_1' \stackrel{\mathrm{def}}{=} R_1 \,|\, P_1\{\tilde{u}/\tilde{x}\}$ and $R_2' \stackrel{\mathrm{def}}{=} R_2 \,|\, P_2\{\tilde{u}/\tilde{x}\}$, since P_1 and P_2 are autonomous, and therefore $P_1\{\tilde{u}/\tilde{x}\}$ and $P_1\{\tilde{u}/\tilde{x}\}$ are, then $R_1', R_2' \in \mathcal{A}_\mathrm{p}$. By Proposition 4-30 and Proposition 4-29, $R_1' \sim_r R_2'$, we have $(Q_1', Q_2') \in \mathcal{R}_1$;

$Q_1 \xrightarrow{{}^{\dagger}m(\tilde{u})} Q_1'$: where ${}^{\dagger}m \neq {}^{\dagger}n$. It is only possible when $R_1 \xrightarrow{{}^{\dagger}m(\tilde{u})} R_1'$ and $Q_1' \equiv R_1' \,|\, !{}^{\dagger}n(\tilde{x}).P_1$. From $R_1 \sim_r R_2$, this implies:

either $R_2 \xrightarrow{{}^{\dagger}m(\tilde{u})} R_2'$, $R_1' \sim_r R_2'$ and $Q_2 \xrightarrow{{}^{\dagger}m(\tilde{u})} Q_2'$ where $Q_2' \equiv R_2' \,|\, !{}^{\dagger}n(\tilde{x}).P_2$. Notice Lemma 4-27, we have $R_1', R_2' \in \mathcal{A}_\mathrm{p}$, therefore $(Q_1', Q_2') \in \mathcal{R}_1$;

or $R_2 \xrightarrow{\tau} R_2'$, $R_1' \sim_r [{}^{\dagger}m\langle\tilde{u}\rangle]R_2'$ and $Q_2 \xrightarrow{\tau} Q_2'$ for $Q_2' \equiv R_2' \,|\, !{}^{\dagger}n(\tilde{x}).P_2$. Again $R_1', R_2' \in \mathcal{A}_\mathrm{p}$ by Lemma 4-27, since $[{}^{\dagger}m\langle\tilde{u}\rangle]Q_2' \equiv [{}^{\dagger}m\langle\tilde{u}\rangle]R_2' \,|\, !{}^{\dagger}n(\tilde{x}).P_2$ according to rule lStr_IND, we have $(Q_1', [{}^{\dagger}m\langle\tilde{u}\rangle]Q_2') \in \mathcal{R}_1$;

$Q_1 \xrightarrow{\alpha} Q_1'$: where α is a not-input action. It is only possible when $R_1 \xrightarrow{\alpha} R_1'$ and $Q_1' \equiv R_1' \,|\, !{}^{\dagger}n(\tilde{x}).P_1$. From $R_1 \sim_r R_2$, this implies $R_2 \xrightarrow{\alpha} R_2'$, $R_1' \sim_r R_2'$ and $Q_2 \xrightarrow{\alpha} Q_2'$ where $Q_2' \equiv R_2' \,|\, !{}^{\dagger}n(\tilde{x}).P_2$, again, with Lemma 4-27, we have $(Q_1', Q_2') \in \mathcal{R}_1$.

Put them together, since all the possible transitions $Q_1 \xrightarrow{\alpha} Q_1'$ are covered by the above cases, by the definition of $\sim_r$ we have $\mathcal{R}_1 \subseteq \sim_r$.

That $\mathcal{R}_2$ is a $\sim_r$ upto $\equiv$ can be shown in a similar way. ■

Proposition 4−35: For autonomous processes, responsive bisimulations are congruences.

Proof: Immediately concluded by the combination of Proposition 4-29, Proposition 4-30, Corollary 4-31, Proposition 4-32, Proposition 4-33, Proposition 4-34 and Corollary 4-23. ■

5 Discussion

5.1 Privatise message versus privatise input port

Some readers may wonder the need for the new term $[{}^{\dagger}m\langle\tilde{u}\rangle]P$; can the same effect be achieved by separating the scope of input and output polars, and configuring ${}^{\dagger}m$ as private? It cannot.

What $[{}^{\dagger}m\langle\tilde{u}\rangle]$ does in the term $[{}^{\dagger}m\langle\tilde{u}\rangle]P$ is to privatise neither polar ${}^{\dagger}m$ nor $\bar{m}$, but the message $\tilde{u}$. The input polar ${}^{\dagger}m$ has not consumed this message yet, and the output polar $\bar{m}$ can remain public so more messages can be sent via it. Especially, any message emitted via $\bar{m}$ by P itself must be considered as part of observation behaviour of $[{}^{\dagger}m\langle\tilde{u}\rangle]P$. The separating of polars' scope has nothing to do with this issue, though may help in describing the concept of "safe process". We chose not to include this separation because this may require introducing another operator, polar matching, which will increase the complexity of expressions rather than simplify them.

We may consider the difference between the term $[{}^{\dagger}m\langle\tilde{u}\rangle]P$ and $\bar{m}\langle\tilde{u}\rangle \mid P$ as that, in the former $[{}^{\dagger}m\langle\tilde{u}\rangle]$ is a buffered message arriving from the channel m and waiting for P or its derivatives to pick it up (but not have to), while in the latter, the $\bar{m}\langle\tilde{u}\rangle$ is an outgoing message to be buffered into the channel m. From an external observer point of view, the sent message $[{}^{\dagger}m\langle\tilde{u}\rangle]$ in the former is invisible, while the message $\bar{m}\langle\tilde{u}\rangle$ in the latter can be mistaken as an output from the target process P. In this sense, we may read $[{}^{\dagger}m\langle\tilde{u}\rangle]P$ as "the behaviour of the black box P while provided with the test message $\tilde{u}$ via channel m", and this behaviour depends on whether and when P or its derivatives are able to access the input port ${}^{\dagger}m$.

For some readers, the role of term $[{}^{\dagger}m\langle\tilde{u}\rangle]P$ can be understood as a weak responsive bisimulation of $(\nu n)(\bar{n}\langle\tilde{u}\rangle \mid !{}^{\dagger}m(\tilde{x}).\bar{n}\langle\tilde{x}\rangle \mid P\{{}^{\dagger}n/{}^{\dagger}m\})$, which is directly concluded from Proposition 3-22. We believe that our choice on introducing the new term $[{}^{\dagger}m\langle\tilde{u}\rangle]P$ gives a clearer and simpler description of semantic than using $(\nu n)(\bar{n}\langle\tilde{u}\rangle \mid !{}^{\dagger}m(\tilde{x}).\bar{n}\langle\tilde{x}\rangle \mid P\{{}^{\dagger}n/{}^{\dagger}m\})$.

The using of input polar ${}^{\dagger}m$ rather than output polar $\bar{m}$ in $[{}^{\dagger}m\langle\tilde{u}\rangle]$ is because that it is a message arriving from channel m rather than being sent to channel m. It is also necessary for preventing an output polar substitution, caused by input prefixing, to change the testing result for the static behaviour of P. Think of it in this way: $[{}^{\dagger}m\langle\tilde{u}\rangle]$ is a sent message, and you cannot change the destination address of mail after it is sent.

5.2 Delay of input versus delay of output

The asynchronous bisimulation in [Amadio96] emphasises the possible delay of message output (or more precisely, delays during the delivery), and considers message retransmission with the same communication channel as ignorable. Its definition written in the polar π-calculus is:

> The (strong) **asynchronous bisimulation** is a (strong) $o\tau$-bisimulation $\mathcal{S}$ for which whenever $P\mathcal{S}Q$ then $P \xrightarrow{{}^{\dagger}m(\tilde{u})} P'$ implies either $Q \xrightarrow{{}^{\dagger}m(\tilde{u})} Q'$ and $P'\mathcal{S}Q'$, or $Q \xrightarrow{\tau} Q'$, and $P'\mathcal{S}(\bar{m}\langle\tilde{u}\rangle \mid Q')$.
>
> The weak version is obtained by replacing transitions with weak transitions everywhere. We denote $\sim_a$ and $\approx_a$ be the largest strong and weak asynchronous bisimulation respectively.

Both the responsive bisimulation and asynchronous bisimulation describe asynchronous communication by allowing message delay. We do not include the asynchronous bisimulation for a couple of reasons:

1. We are interested in the delay of input rather than that of output;
2. To capture the delay of output, the asynchronous bisimulation allows competition on grabbing messages for the same input port, which can disturb the detection of responsive behaviours;
3. Combining both output delay and input delay will make the theory unnecessarily complicated.

In contrary, the responsive bisimulation concentrates on the delay of input. In the view of object-oriented programming, the delay in the delivery is not visible for either sender or receiver, and is also out of their control. The delay of input, however, is controllable for the receiver, and, as pointed out by [McHale94] and [Zhang98B], the existence of the interval between the event of a message arriving at an object and the event of the message processing starting, provides a synchronisation control point for concurrent objects. In other words, the responsive bisimulation is quite natural for compositional objects.

The asynchronous bisimulation and the responsive bisimulation overlap, but neither contains the other. For example, given $m \notin fin(P)$, then the processes $\overset{+}{m}.\bar{m} \mid P$ and P are clearly weakly asynchronous bisimilar, but not weakly responsive bisimilar, while the processes $(\nu n)(\overset{+}{m}.\bar{n} \mid \overset{+}{k}.\overset{+}{n}.P)$ and $\overset{+}{k}.\overset{+}{m}.P$ are clearly weakly responsive but not asynchronous bisimilar. The result given by Proposition 3-22, as a counterpart of the asynchronous bisimulation conclustion $\overset{+}{m}.\bar{m} \mid P \approx_{a} P$, is another example where weakly responsive bisimilar does not agree with weakly asynchronous bisimulation.

It is also worth to noting that the 1-bisimulation, which was proven to be equivalent to the asynchronous bisimulation and defined as “an oτ-bisimulation $\mathcal{S}$ where $P\mathcal{S}Q$ implies $(P \mid \bar{m}\langle\tilde{u}\rangle)\mathcal{S}(Q \mid \bar{m}\langle\tilde{u}\rangle)$ for all $\bar{m}\langle\tilde{u}\rangle$” in [Amadio96], becomes irrelevant in the polar π-calculus, because an unbound synchronisation is no longer considered as a τ action, as discussed in Remark 2-3. Apart from this, the major difference between $\sim_1$ and $\sim_{r1}$, which has some structural similarity, is that, $\sim_{r1}$ examines how processes respond to various inputs by filtering out the effects of the environment's behaviour, whereas $\sim_1$ treats processes as part of the environment during the testing.

5.3 Message localisation verses non-blocking input prefix

Though the localisation term $[\overset{+}{m}\langle\tilde{u}\rangle]P$ seem not fimiliar for most of readers, there is a comparable concept. The notion of non-block input prefix was introduced by [Parrow97] and [Victor98], and was adopted by [Merro98] where it is also called “delayed input prefix” for a term $m(\tilde{x})P$. While there are some similarity in the structure of their inference rules, the concepts are different. First, the box $[\overset{+}{m}\langle\tilde{u}\rangle]$ in $[\overset{+}{m}\langle\tilde{u}\rangle]P$ represents a particular message buffered in channel m, and $\tilde{u}$ are free names in $[\overset{+}{m}\langle\tilde{u}\rangle]P$, and P does not have to have the ability to consume this message. In contrast, $m(\tilde{x})P$ indicates the potential ability of inputting any message from channel m while it does not prevent P to perform other actions which are not along bound names $\tilde{x}$. Second, $[\overset{+}{m}\langle\tilde{u}\rangle]P$ describing an envirnment state for testing the responsive behaviour of process P, while $m(\tilde{x})P$ tries to model a process behaviour itself.

5.4 Porcess safety verse name hidden

Since the responsive bisimulation is mostly useful only for safe processes, which own the receptors they use, a question arises: can the same effect of the responsive bisimulation be achieved by limiting which names or polars can be visitable in a bisimulation relation? We are not seeking such an approach because the following difficulties, among many others:

1. A bisimulation relation associating with certain names is not useful in comparing processes behaviour in generic. For example, to inference the behaviour similarity between $P \mid R$ and $Q \mid R$ from that between P and Q, the names associated may have to be changed, that is, they cannot be measured under the same kind of relation.
2. The both sets of the names which a process own and does not own the input polar are dynamic and infinite during the course of reduction. For example, the reduction from $(\nu n)(P \mid \bar{m}\langle\bar{n}\rangle) \xrightarrow{(\nu n)\bar{m}\langle\bar{n}\rangle} P$ may add a new public name n, to the owned name set.
3. An output action $\bar{m}\langle\tilde{u}\rangle$, performed by the target process should always be observed for determining the responsive behaviour, no matter whether the target process own the input polar $\bar{m}$ or not. Therefore no such a name m or polar $\bar{m}$ should be hidden.

5.5 Relation of the responsive bisimulation with some conventional bisimulations

Since the polar π-calculus is a subcalculus of the asynchronous π-calculus, generic properties concluded from the full domain of the asynchronous π-calculus can also apply to the polar π-calculus in its shrunken domain. For example, the ground bisimulation, early, late and open bisimulations all coincide in the asynchronous π-calculus, as well as in the polar π-calculus. The asynchronous π-calculus itself, as pointed out in [Amadio96], is a subcalculus of the standard π-calculus, with the restrictions that outputs cannot be used as prefix or on a choice point. Therefore, generic properties concluded from the standard π-calculus can also apply to the asynchronous π-calculus and the polar π-calculus.

One of the conclusions from [Amadio96] is that, the ground, early, late and open bisimulations all coincide in the asynchronous π-calculus. Similar results can also be concluded for the polar π-calculus.

To keep the syntactical consistence, we redefine those similar bisimulation relations in the polar π-calculus.

Definition 5−36: The (strong) **ground bisimulation** is a (strong) oτ-bisimulation $\mathcal{S}$ if whenever $P\mathcal{S}Q$ then $P \xrightarrow{\dagger m(\tilde{u})} P'$ implies either $Q \xrightarrow{\dagger m(\tilde{u})} Q'$. And the weak ground bisimulation is obtained by replacing transitions with weak transitions everywhere. We denote $\sim_g$ and $\approx_g$ be the largest strong and weak ground bisimulation respectively. Clearly, $\sim_g \subseteq \approx_g$.

This definition has adopted the ground style of [Sangiorgi95], that is, no name substitution is needed in the input clause.

Lemma 5−37: ground bisimulations are preserved by output polarity name substitution.
Proof: Similar to that for Proposition 4-30, except no need to check the cases involving localisation. ■

Definition 5−38: The (strong) **early bisimulation** is a strong oτ-bisimulation $\mathcal{S}$ if whenever $P\mathcal{S}Q$ then $P \xrightarrow{\dagger m(\tilde{u})} P'$ implies $\forall \tilde{y}\ \exists Q'$ s.t. $Q \xrightarrow{\dagger m(\tilde{u})} Q'$ and $P'\{\tilde{y}/\tilde{u}\}\mathcal{S}\, Q'\{\tilde{y}/\tilde{u}\}$;

The (strong) **late bisimulation** is a strong oτ-bisimulation $\mathcal{S}$ if whenever $P\mathcal{S}Q$ then $P \xrightarrow{\dagger m(\tilde{u})} P'$ implies $\exists Q'$ s.t. $Q \xrightarrow{\dagger m(\tilde{u})} Q'$ and $\forall \tilde{y}$ $(P'\{\tilde{y}/\tilde{u}\}\mathcal{S}\, Q'\{\tilde{y}/\tilde{u}\})$;

The (strong) **open bisimulation** is a strong oτ-bisimulation $\mathcal{S}$ if whenever $P\mathcal{S}Q$ then for any output polar substitution $\sigma=\{\tilde{y}/\tilde{x}\}$, $P\sigma \xrightarrow{\alpha} P'$ implies $\exists Q'$ s.t. $Q\sigma \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$;

The weak versions of those bisimulations are obtained by replacing transitions with weak transitions everywhere. We denote the largest strong early, late and open bisimulations with $\sim_e$, $\sim_l$ and $\sim_o$ respectively, and denote the largest weak early, late and open bisimulation with $\approx_e$, $\approx_l$ and $\approx_o$ respectively.

Lemma 5−39: Ground, early, late and open bisimulations all coincide.
Proof: The proof is trivial. Let $\mathcal{S}_g$ be strong (or weak) ground bisimulation, Let $\mathcal{S}$ be any of strong (or weak, respectively) early bisimulation, late bisimulation or open bisimulation, then we got

$P\mathcal{S}Q$ implies $P\mathcal{S}_gQ$, by letting $\tilde{y}=\tilde{u}$; $P\mathcal{S}_gQ$ implies $P\mathcal{S}_gQ$, by applying Lemma 5-37. ■

Corollary 5−40: The ground bisimulations are responsive bisimulations, that is, $\sim_g \subseteq \sim_r$ and $\approx_g \subseteq \approx_r$.
Proof: Directly concluded from the comparison of their definitions.

The Figure 5-1 shows the relations between some bisimulations in the polar π-calculus, where the arrow means "is a subset of".

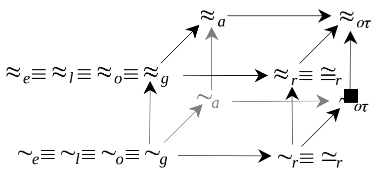


Figure 5-1

6 Application

With the responsive bisimulation, behavioural equivalence can be recovered for compositional objects. As already pointed out, the objects O_1 and O_2 of Figure 1-1 are behaviourally the same in the client's eyes, which now can be expressed as $O_1 \approx_r O_2$. Also, for the mailroom example, whether a tenant is "good" or "bad" will be no longer related to where his mailbox is located.

Generally, with the idea of [Zhang98A], let I be the index set, $I_1, I_2, \ldots, I_n$ be disjoint subsets of I where $I_1 \cup I_2 \cup \ldots \cup I_n \equiv I$, then the functional behaviour of a concurrent object and a control process can be modelled in the polar π-calculus respectively as

$F \stackrel{\text{def}}{=} (\overset{\dagger}{\tilde{m}}) \prod_{i \in I} !\overset{\dagger}{m}_i(\tilde{x}).M_i$, where M_i represents the body of the i$^{\text{th}}$ method;
$C \stackrel{\text{def}}{=} (\overset{\dagger}{\tilde{n}}, \bar{\tilde{m}})(C_1 | C_2 | \ldots | C_n)$, where each C_k has the form of either $C_k \stackrel{\text{def}}{=} \sum_{i \in I_k} \overset{\dagger}{n}_i(\tilde{x}).\bar{m}_i\langle\tilde{x}\rangle$ or $C_k \stackrel{\text{def}}{=} \prod_{i \in I_k} !\overset{\dagger}{n}_i(\tilde{x}).\bar{m}_i\langle\tilde{x}\rangle$.

Then when composing C with F, we have $(\nu\, \tilde{m})(C\langle\overset{\dagger}{\tilde{n}}, \bar{\tilde{m}}\rangle | F\langle\overset{\dagger}{\tilde{m}}\rangle) \approx_r R$ where

$R \stackrel{\text{def}}{=} R_1 | R_2 | \ldots | R_n$, with each R_k has the form of either $R_k \stackrel{\text{def}}{=} \sum_{i \in I_k} \overset{\dagger}{n}_i(\tilde{x}).M_i$ or $R_k \stackrel{\text{def}}{=} \prod_{i \in I_k} !\overset{\dagger}{n}_i(\tilde{x}).M_i$.

In other words, the control process C defines the exclusion behaviour for methods separately from the functional behaviour F, and both are enforced in the composed object.

The above is only a simplified description. In the more sophisticated model ([Zhang98A], [Zhang01C]), a scheduler for each method is contained in the unified form of control elements: $C_k \stackrel{\text{def}}{=} \sum \overset{\dagger}{n}_i(\bar{s}_n, \bar{r}_n, \bar{t}_n, \bar{\tilde{v}}).(\nu\, s_m, r_m, t_m) S_i$, where $\bar{\tilde{v}}$ are the parameters (i.e., the message) of the function call, s,r,t are signal channels for the synchronisation points during the method execution: start, value return, and termination respectively.The follows are three scheduler examples:

$S_i' \stackrel{\text{def}}{=} \bar{s}_n | \bar{m}_i\langle \bar{s}_f, \bar{s}_m, \bar{r}_m, \bar{t}_m, \bar{\tilde{v}}\rangle | \overset{\dagger}{s}_m.\overset{\dagger}{r}_m(\tilde{u}).\overset{\dagger}{t}_m.(\bar{r}_n\langle\tilde{u}\rangle | \bar{t}_n | C_k))$ -- Mutually exclusive;
$S_i'' \stackrel{\text{def}}{=} \bar{s}_n | \bar{m}_i\langle \bar{s}_f, \bar{s}_m, \bar{r}_m, \bar{t}_m, \bar{\tilde{v}}\rangle | \overset{\dagger}{s}_m.\overset{\dagger}{r}_m(\tilde{u}).(\bar{r}_n\langle\tilde{u}\rangle | \overset{\dagger}{t}_m.(\bar{t}_n | C_k)))$ -- Mutually exclusive with early return;
$S_i''' \stackrel{\text{def}}{=} \bar{s}_n | C_k | \bar{m}_i\langle \bar{s}_f, \bar{s}_m, \bar{r}_m, \bar{t}_m, \bar{\tilde{v}}\rangle | \overset{\dagger}{s}_m.(\overset{\dagger}{r}_m(\tilde{u}).\bar{r}_n\langle\tilde{u}\rangle | \overset{\dagger}{t}_m.\bar{t}_n))$ -- Non-exclusive, non-constraint.

Here the role of C_k in the S_i expressions can be regarded as the "unlock point". [Zhang01C] and [Zhang01D] have shown that, from the unlock scheduling point of view, the number of S_i types is finite, and the composition effects can also be grouped to a finite number of types, which can be useful for compile time reasoning and code optimisation.

In [Zhang01C] and [Zhang01D] the concurrent object model is described using the κ-calculus ([Zhang01A]), which is much more expressive and flexible on behaviours composition/separation. Also, more complicated controls can be described in the same unified form, andunlock points C_k will no longer have to appear in S_i expression explicitly. However, this is out of the scope of this paper.

To investigate the properties of object composition further, we need some more terminologies and symbols.

Definition 6−41: A safe process P is an *object component process* with *source set* $\overset{\dagger}{\tilde{m}}$, where $\tilde{m} = \mathit{fin}(P)$, if $P \downarrow \overset{\dagger}{m}_i$ for all $m_i \in \tilde{m}$.

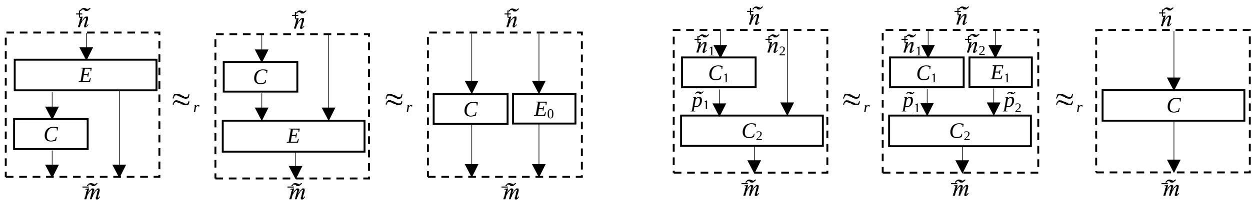
The object component process C with source set $\overset{\dagger}{\tilde{n}}$ is a *control process* with *socket set* $\overset{\dagger}{\tilde{n}}$ and *plug set* $\bar{\tilde{m}}$, if $\tilde{m} \subseteq \mathit{fon}(C)$, $\tilde{m} \cap \tilde{n} = \varnothing$, and for each i where $\overset{\dagger}{n}_i \in \overset{\dagger}{\tilde{n}}$ and $\bar{m}_i \in \bar{\tilde{m}}$, there exists some processes C', C'' and action sequence ℓ satisfying $\{\tilde{n}, \tilde{m}\} \cap \mathit{fin}(\ell) = \varnothing$, such that $C\langle\overset{\dagger}{\tilde{n}}, \bar{\tilde{m}}\rangle \xrightarrow{\overset{\dagger}{n}_i(\tilde{u})} C'$, $C' \xrightarrow{\ell} C''$ and $C'' \downarrow \bar{m}_i\langle\tilde{u}\rangle$.

We define the generic empty control process as $E \stackrel{\text{def}}{=} (\overset{\dagger}{\tilde{n}}, \bar{\tilde{m}}) \prod_{i \in I} !\overset{\dagger}{n}_i(\tilde{x}).\bar{m}_i\langle\tilde{x}\rangle$

Given a control process D with socket set $\overset{\dagger}{\tilde{x}}$ and plug set $\bar{\tilde{y}}$ and an object component process Q with source set $\overset{\dagger}{\tilde{z}}$, let $C \stackrel{\text{def}}{=} (\overset{\dagger}{\tilde{x}}, \bar{\tilde{y}})D$ and $P \stackrel{\text{def}}{=} (\overset{\dagger}{\tilde{z}})Q$, then we denote $(\tilde{m})(\nu\, \tilde{n})(C\langle\overset{\dagger}{\tilde{m}}, \bar{\tilde{n}}\rangle | P\langle\tilde{n}\rangle)$ with the abbreviation $C \succ P$, for all $\tilde{m}$ and $\tilde{n}$ where $\{\tilde{m}, \tilde{n}\} \cap \mathit{fn}(D) = \varnothing$ and $\{\tilde{m}, \tilde{n}\} \cap \mathit{fn}(Q) = \varnothing$.

One of the desired properties of the composition is the identity law. With ground bisimulation, [Zhang98A] has proven the identity law on the right $C \succ E \approx_g C$, but the left identity law ($E \succ C \approx_g C$) is not generally true. With the responsive bisimulation, however, the identity law holds for both sides: $E \succ C \approx_r C \succ E \approx_r C$, proven by [Zhang01C]. This property not only gives mathematical elegance, or reflects the fact that adding an empty behaviour to a server object will make no difference in the clients' eyes, but more importantly, it means that we can always add new constrains to the existing control with relatively simple composition, without introducing unexpected side effect in behaviour. For example, assume the control process C_1 describes and only describes the exclusion between $\tilde{m}_1$ and $\tilde{m}_2$, and the control process C_2 describes and only describes the exclusion between $\tilde{m}_2$ and $\tilde{m}_3$, then $C_1 \succ C_2$ will provider both exclusion between $\tilde{m}_1$ and $\tilde{m}_2$, and that between $\tilde{m}_2$ and $\tilde{m}_3$, but no other exclusion will be accidentally added or removed.

Figure 6-1 shows some more examples using the identity law in compositional object modelling. The example shown in the left diagram indicates that the same effect of this control can be constructed in three different ways: using the empty control E to extend the scope of controller C to $\tilde{m}$, adding the constraint described by C to the empty control E, using two independent controllers C and E_0.



$E \succ C \approx_r C \succ E \approx_r C \mid E_0$

$(C_1\langle \tilde{n}_1,\tilde{p}_1\rangle \succ E_1\langle \tilde{n}_2,\tilde{p}_2\rangle) \succ C_2\langle \tilde{p},\tilde{m}\rangle \approx_r C\langle \tilde{n},\tilde{m}\rangle$

where $\tilde{n}_2 = \tilde{n} - \tilde{n}_1$, $\tilde{p} = \tilde{p}_1 \cup \tilde{p}_2$.

Figure 6-1

Another proven property of composition is the association law, held by both ground bisimulation ([Zhang98A]) and the responsive bisimulation ([Zhang01C]), that is: $(C_1 \succ C_2) \succ C_3 \approx_g C_1 \succ (C_2 \succ C_3)$ and $(C_1 \succ C_2) \succ C_3 \approx_r C_1 \succ (C_2 \succ C_3)$.

7 Conclusion

This paper has presented the responsive bisimulation, which can capture responsive behavioural equivalence between compositional concurrent objects, by allowing the delay of input actions. For object systems, where input name clash can be eliminated, the responsive bisimulations are preserved by parallel composition, output name substitution and choice, and can even be congruence.

The responsive bisimulation can be understood in different ways. Apart from the view of "input delay", another view is that, when testing the behaviour of the target object, or black box, the only precondition we need to know is what messages have been provided to it, and the only postcondition we should examine is the response from the target object. We have proven these different views are equivalent.

With the responsive bisimulation, we can have a broader and more generic study of the behaviour of concurrent components, where existing bisimulations fail to give us the desired equivalence. Our approach enables us to establish a theory of concurrent objects with elegant compositional properties and provides a semantic basis for an extension to concurrent object-oriented programming languages.

References:

[Aksit92] Mehmet Aksit and Lodewijk Bergmans "Obstacles in Object-oriented Software Development", *OOPSLA'92 Conference Proceedings*, volume 27 of ACM SIGPLAN Notices, pages 341-358, New York, October 1992

[Amadio96] Roberto M. Amadio, Ilaria Castellani and Davide Sangiorgi, "On Bisimulations for the Asynchronous π-calculus", in *Proceedings of CONCUR'96*, LNCS volume 1119, Springer Verlag, 1996

[Holmes97] David Holmes, James Noble, John Potter, "Aspects of Synchronisation", in Christine Mingins, Roger Duke and Bertrand Meyer, editors, *Technology of Object-Oriented Languages and Systems TOOLS 25 - Proceedings of The 25th International Conference TOOLS* (TOOLS Pacific'97), pages 7-18, Melbourne, Australia, November 1997.

[Honda91] Kohei Honda and Mario Tokoro, "An Object Calculus for Asynchronous Communication", in P. America, editor, *ECOOP'91*, LNCS vol 512, pages 133-147, Springer-Verlag, 1991.

[Hüttel96] Hans Hüttel and Josva Kleist, "*Objects as mobile processes*", Aalborg University, August 1996.

[Jones93] Cliff B. Jones, "A π-calculus Semantics for an Object-based Design Notation", in E. Best, editor, *Proceedings of CONCUR'93*, volume 715 of *Lecture Notes in computer Science*, pages 158-172. Springer Verlag, 1993

[McHale94] Ciaran McHale, "Synchronisation in Concurrent, Object-oriented Languages: Expressive Power, Genericity and Inheritance", PhD. Thesis, Department of Computer Science, Trinity college, University of Dublin, Ireland, October 1994.

[Merro98] Massimo Merro and Davide Sangiorgi, "On Asynchrony in Name-passing calculi", In 25th ICALP, volune 1443 of *Lecture Notes in computer Science*, pages 856-867. Springer Verlag, 1998.

[Merro00] Massimo Merro, Josva Kleist and Uwe Nestmann, "Local π-Calculus at Work: Mobile Object as Mobile Processes", In *Proc. of IFIP TCS2000*, Sendai, Japan, Aug. 2000. LNCS vol. 1872, pages 390-408, Springe.

[Milner92] Robin Milner, Joachim Parrow, David Walker, "A Calculus of Mobile Process" (Parts I and II), *Journal of Information and Computation*, 100:1-77, September 1992.

[Milner92b] Robin Milner and Davide Sangiorgi, "Barbed Bisimulation", in W. Kuich, editor, *Proceeding of 19th ICALP*, volume 623 of *Lecture Notes in computer Science*, Springer Verlag, 1992

[Milner99] Robin Milner, "*Communicating and Mobile Systems: the π-calculus*", Cambridge University Press,1999

[Noble00] James Noble and John Potter, "Exclusion for Composite Objects", In *Proceedings of OOPSLA 2000*, Minneapolis, Minnesota USA, ACM press, 2000

[Odersky95a] Martin Odersky, "Polarized Name Passing", in Proceedings of 15th Foundations of Software Technology and Theoretical Computer Science (FST&TCS'95), Bangalore, India, December 18-20, 1995. URL: http://lampwww.epfl.ch/~odersky/papers

[Odersky95c] Odersky, M. "Polarized bisimulation", In *Proceedings of Workshop on Logic, Domains, and Programming Languages*, Darmstadt, Germany, 1995

[Parrow97] Joachim Parrow and Björn Victor, "The Update Calculus", In Michael Johnson, editor, *Proceedings of AMAST'97*, Sydney, Australia, Dec. 13-17th 1997, LNCS 1349, pages 409-423, Springer 1997

[Ravara97] António Ravara and Vasco T. Vasconcelos, Behavioural types for a calculus of concurrent objects. In C. Lengauer, M. Griebl, and S. Gorlatch, editors, *Proceedings of 3rd International Euro-Par Conference*, LNCS 1300, pages 554--561. Springer-Verlag, 1997

[Sangiorgi92b] Davide Sangiorgi, "*Expressing Mobility in Process Algebras: First-Order and Higher-Order paradigms*", PhD thesis, Computer Science Department, University of Edinburgh, UK, 1992.

[Sangiorgi95] David Sangiorgi, "*Lazy functions and mobile processes*", INRIA Technical Report RR-2515, August 1996.

[Sangiorgi96] Davide Sangiorgi, "*An Interpretation of Typed Objects into Typed π-calculus*", INRIA Technical Report RR-3000, August 1996.

[Sangiorgi96b] Davide Sangiorgi, "Locality and Non-interleaving Semanitics in Calculi for Mobiule Processes", *Theoretical Computer Science*, 155:39-83, 1996

[Schneider97] Jean-guy Schneider and Markus Lumpe, "Synchronizing Concurrent Objects in the Pi-Calculus", Proceedings of Langages et Modèles à Objets '97, Roland Ducournau and Serge Garlatti (Ed.), pp.61-76, Hermes, Roscoff, October 1997.

[Victor98] Björn Victor, "*The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes*", PhD thesis, Dept. of Computer Systems, Uppsala University, Sweden, June 1998

[Walker95] David Walker, "Objects in the π-Calculus", *Information and Computation*, 116(2):253-271, 1995

[Zhang97] Xiaogang Zhang and John Potter, "Class-based models in π-calculus", in Christine Mingins, Roger Duke and Bertrand Meyer, editors, *Technology of Object-Oriented Languages and Systems*, *TOOLS 25* (TOOLS Pacific'97), Melbourne, Australia, 24th-27th November 1997, pages 238-251, IEEE Computing Society Press, 1998.

[Zhang98A] Xiaogang Zhang and John Potter, "*Compositional Concurrency Constraints for Object Models in π-calculus*", Technical Report C/TR-9804, Macquarie University, Sydney, Australia, 1998.

[Zhang98B] Xiaogang Zhang and John Potter, "A Composition Approach to Concurrent Objects", in Jian Chen, Mingshu Li, Christine Mingins and Bertrand Meyer, editors, *Technology of Object-Oriented Languages and Systems*, *TOOLS 27* (TOOLS Asia'98), Beijing, China, 22nd-25th September 1998, pages 116-126, IEEE Computing Society Press, 1998.

[Zhang01A] Xiaogang Zhang and John Potter, "*A Constraint Description Calculus for Compositional Concurrent Objects*", Technical report UNSW-CSE-TR-0204.

[Zhang01B] Xiaogang Zhang and John Potter, "*On Responsive Bisimulations in the κ-calculus*", Technical report UNSW-CSE-TR-0205, 2002.

[Zhang01C] Xiaogang Zhang and John Potter, "*Compositional Concurrent Objects*", in preparation.

[Zhang01D] Xiaogang Zhang and John Potter, "A *Compositional Concurrent Object Model, -- From Theory to Practise*", in preparation.