A Model of Service-Oriented Computation

Luis Caires, Hugo T. Vieira, J. C. Seco

CITI / Departamento de Informática, Universidade Nova de Lisboa, Portugal

October 31, 2007
(revised March 18, 2008)

Abstract

We propose a motivation from "first principles" of a small set of abstractions for expressing and analyzing service based systems. In particular, we suggest the aspects of distribution, process delegation, communication and context sensitiveness, and loose coupling as specific to the service-oriented computational model. Distinguishing aspects of our model are the adoption of a very simple, context sensitive, message-passing local communication mechanism, and a novel mechanism for handling exceptional behavior. We instantiate our model by extending of a fragment of the π-calculus, and show that behavioral equivalences, defined by strong bisimilarity and weak bisimilarity, are congruences for all operators.

1 Introduction

Web services have emerged mainly as a toolkit of technological and methodological solutions for building open-ended collaborative software systems on the internet. Main actors behind this impulse include top software companies, some of them acting quite independently, and industrial consortia such as the W3C. Many concepts that are frequently put forward as distinctive of service-oriented computing, namely, object-oriented distributed programming, long duration transactions and compensations, separation of workflow from service instances, late binding and discovery of functionalities, are certainly not new, at least when considered in isolation.

What is certainly new about services is that they are contributing to physically realize (on the internet) a global, interaction-based, loosely coupled, model of computation. Although this transition may look for many IT actors as leading to unknown territory, it opens challenging opportunities for the Computer Science research communities that have been studying general interaction-based models of computation (e.g., process algebras) and their properties for decades. In fact, one would like to better understand in what sense service orientation is to be seen as a new paradigm to build and reason about distributed systems. Of course, the global computing infrastructure is bound to remain highly heterogeneous and dynamic in its capabilities and phenomena, so it does not seem reasonable to foresee the premature emergence of comprehensive theories and technological artifacts, well suited for everyone and every application. This suggests that one should focus not only on particular systems and theories themselves, but also on general systems, properties, and their interfaces.

In this report, we propose a new model of service-oriented computation. Our starting point is an attempt to isolate and clarify essential characteristics of the service-oriented model of computation, in order to propose a motivation from "first principles" of a reduced set of general abstractions for expressing and analyzing service based systems. To focus on a set of independent primitives, we instantiate our model by modularly extending the static fragment of the

π-calculus with a notion of conversation context, message-passing communication primitives, and an exception handling mechanism.

We have tried to keep our realization fairly general (hoping to escape from the doom of premature optimization or specificity, and wishing to provide a wide basis for many different kinds of analysis and many different techniques), to achieve some level of simplicity and clarity, and to ensure orthogonality and semantic independence of the primitives (so that we may easily consider fragments).

Concerning this last point, we may show that strong and weak bisimilarity, defined in the standard way, are congruences for all operators of the language, this fact justifies the semantic status as behavioral operators of the various primitives considered.

2 Aspects of Services

In this section, we attempt to identify some essential characteristics of the service-oriented model of computation, in order to propose a motivation from "first principles" of a reduced set of general abstractions for expressing and analyzing service based systems. We assume some familiarity with concepts and technological artifacts of the "services world".

We identify as key aspects of the service-oriented computational model: *distribution*, process *delegation*, communication and *context* sensitiveness, and *loose coupling*.

Distribution

The purpose of a service relationship is to allow the incorporation of certain activities in a given system, without having to engage *local* resources and capabilities to support or implement such activities. By delegating activities to an external service provider, which will perform them using its own *remote* resources and capabilities, a computing system may concentrate on those tasks for which it may autonomously provide convenient solutions. Thus, the notion of service makes particular sense when the service provider and the service client are separate entities, with access to separate sets of resources and capabilities. This understanding of the service relationship between provider and client assumes an underlying distributed computational model, where client and server are located at least in distinct (operating system) processes, more frequently in distinct sites of a network.

The invocation of a service by a client results in the creation of a new service instance. A service instance is composed by a pair of endpoints, one endpoint located in the server site, where the service is defined, the other endpoint in the client site, where the request for instantiation took place. From the viewpoint of each partner, the respective endpoint acts as a local process, with potential direct access to local resources and capabilities. Thus, for us an endpoint is not a name, a port address, or channel, but an interactive process. Both endpoints work together in a tightly coordinated way, by exchanging data and control information through a hidden communication tunnel (just known by the endpoints).

An essential concept arising is then the notion of "remote process delegation". Remote process delegation differs from the more restricted notion of "remote operation invocation", as known, e.g., from distributed object systems, or even from earlier client-server or remote procedure call based systems. We elaborate on this point in the next item.

Process Delegation versus Operation Invocation

The primitive remote communication mechanism in distributed computing is message passing, yielding asynchronous interaction.

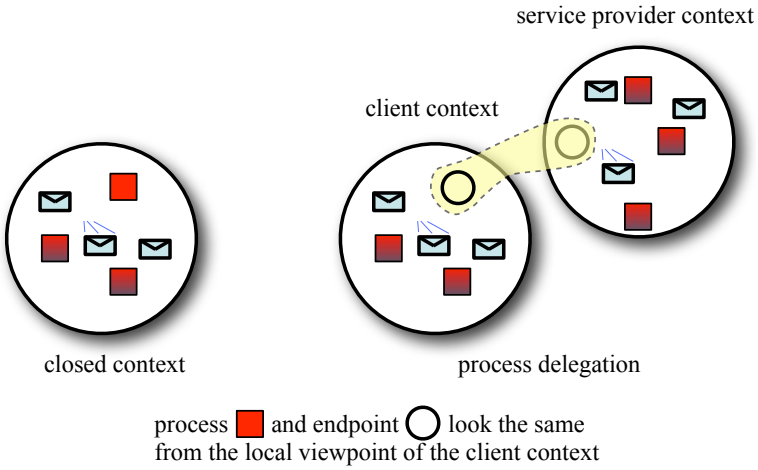

Figure 1: Process delegation.

On top of this basic mechanism, the only one actually implementable, more sophisticated abstractions may be represented, namely remote procedure call (passing first order data) and remote method invocation (also passing remote object references). Along these lines, we see service invocation as a still higher level mechanism, allowing the service client to delegate to a remote server not just a single operation or task, but the execution of a whole interactive activity (technically, a process). By invoking a service, and thus creating a new service instance, a client incorporates in its workflow a process (a dynamic stateful interactive entity) that, although executing remotely in the server environment, appears to the client as a local subsystem. As typical examples of subsystems we may think of a service to book flights (cf., a travel agency), a service to order goods (cf., a purchasing department), a service to process banking operations (cf., an ATM), a service to store and retrieve documents (cf., an archive), a service to receive and send mail (cf. an expedition service), and so on. The distinguishing feature of service-oriented computing, in our view, is an emphasis on the remote delegation of *interactive processes*, rather than on the remote delegation of individual operations.

The remote process delegation paradigm seems more general than the remote operation invocation paradigm, at least at our current level of description, since one can always see an individual operation as a special case of an interactive process. Conversely, one may argue that the delegation of a given interactive process may always be implemented, at a lower level, by the invocation of individual remote operations. However, our view of remote process delegation makes special sense when the endpoints of a service instance make essential use of being located in different execution contexts.

Communication, Contexts, and Context Sensitiveness

The endpoints of a particular service instance are created in different spatial contexts: typically one endpoint will be located in the client context, and the other endpoint in the service provider context. Each endpoint apears to its surrounding context as a local process, even if it offers a communication path to its (dual) remote endpoint.

For example, consider the scenario where the endpoint realizing an archiving functionality in the client context communicates with the other subsystems of the client, e.g., to receive document archiving requests and document retrieval requests, while the corresponding remote endpoint in the server site will communicate with other subsystems in the service provider con-

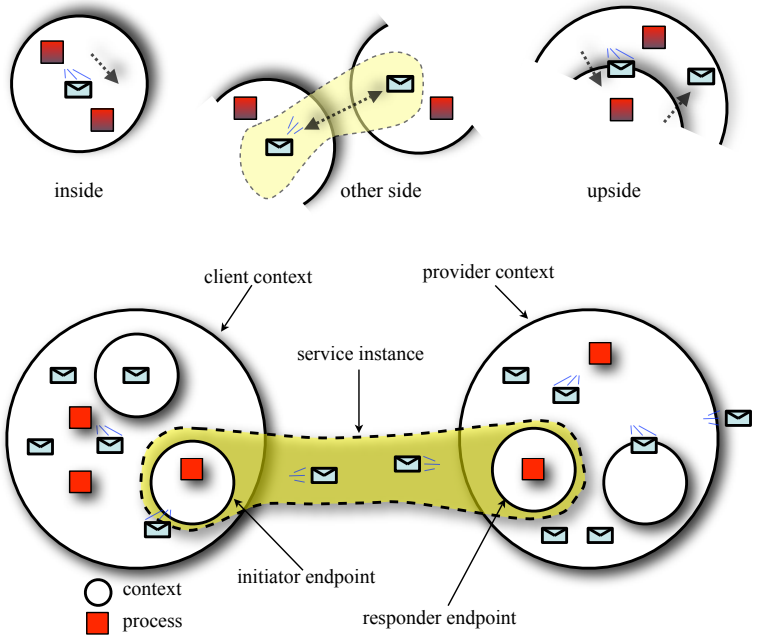

Figure 2: Contexts and Communication Pathways.

text, e.g., the database, the indexing infrastructure, and other resources needed for the mission of the service provider.

We understand an endpoint just as a particular case of a delimited context or scope of interaction. More generally, a context is a delimited space were computation and communication happens. A context may have a spatial meaning, e.g., as a *site* in a distributed system, but also a behavioral meaning, e.g., as *context of conversation*, or a *session*, between two or more partners. For example, the same message may appear in two different contexts, with different meanings (web services technology has introduced artifacts such as "correlation" to determine the appropriate context for otherwise indistinguishable messages).

Thus, the notion of context of conversation seems to be a convenient abstraction mechanism to structure the several service instances collaborating in a service-oriented system.

Computations interacting in a context may offer essentially three forms of communication capabilities. First, they may communicate inside the context, this would correspond to regular internal computations of the context. Second, an endpoint must be able to send messages to and receive messages from the other (dual) endpoint of the context, reflecting interactions between the client and the server roles of a service instance. Third, it must be able to send messages to and receive messages from the enclosing context. Contexts as the one described may be nested at many levels, corresponding to subsidiary service instances, processes, etc.

Notice that we do not expect communication to happen between arbitrary contexts, but rather to always fall in one the three special cases described above: internal interaction (inside the given context), remote interaction (with the dual endpoint), and external interaction (with the immediatly external context).

In Figure 2 we illustrate our intended context dependent communication model, and the various forms of interaction it admits.

We expect that while service invocation may be sensibly performed across any contexts (at an arbitrary "distance"), by any process in possession of the service public name (cf., the URI of the service), any message-based interactions between different conversation contexts should

only be allowed across a single boundary at most, and when the involved contexts are somehow (co)related.

A context is also a natural abstraction to group and publish together closely related services. Typically, services published by the same entity are expected to share common resources, we notice that such sharing is common at several scales of granularity. Extreme examples are an object, where the service definitions are the methods and the shared context the object internal state, and an entity such as, e.g., Amazon, that publishes several services for many different purposes; such services certainly share many internal resources of the Amazon context, such as databases, payment gateways, and so on.

Delimited contexts are also natural candidates for typing, in terms of the messages interchange patterns that may happen at its border. We would thus expect types (or logical formulas) specifying various properties of interfaces, of service contracts, of endpoint session protocols, of security policies, of resource usage, and of service level agreements, to be in general assigned to context boundaries. Enforcing boundaries between subsystems is also instrumental to achieve loose coupling of systems.

Loose Coupling

A service based computation usually consists in an collection of remote partner service instances, in which functionality is to be delegated, some locally implemented processes, and one or more control (or orchestration) processes.The flexibility and openness of a service based design, or at least an aimed feature, results from the loose coupling between these various ingredients. For instance, an orchestration describing a "business process", should be specified in a quite independent way of the particular subsidiary service instances used, paving the way for dynamic binding and dynamic discovery of partner service providers. In the orchestration language `WSBPEL` [1], loose coupling to external services is enforced to some extent by the separate declaration of "partner links" and "partner roles" in processes. In the modeling language SRML [9], the binding between service providers and clients is mediated by "wires", which describe plugging constraints between otherwise hard to match interfaces.

To avoid tight coupling of services, the interface between a service instance (at each of its several endpoints) and the context of instantiation should in general be mediated by appropriate mediating processes, in order to hide and/or adapt the end-point communication protocols (which is in some sense dependent of the particular implementation or service provider chosen) to the abstract behavioral interface expected by the context of instantiation. All computational entities cooperating in a service task should then be encapsulated (delimited inside a conversation context), and able to communicate between themselves and the outer context only via some general message passing mechanism.

Other Aspects

There are many other aspects that must be addressed in a general model of service-oriented computation. The most obvious ones include of course, failure handling and resource disposal, security (in particular access control, authentication and secrecy), time awareness, and a clean mechanism of interoperation. This last aspect seems particularly relevant, and certainly suggests an important evaluation criteria for any service-oriented computation model.

Non Characteristics of Services

We mention here some interesting features of distributed systems that seem fairly alien to the service-oriented computational model. Forms of code migration (weak mobility) seem to re-

quire an homogeneous execution support infrastructure, and thus to run against the aim to get loose coupling, openness and independent evolution of subsystems. In general, any mechanism relying on centralized control or authority mechanisms, or that require a substantial degree on homogeneity in the runtime infrastructure, for example strong mobility, also seem hard to accommodate.

3 The Conversation Calculus

In this section, we motivate and present in detail the primitives of our calculus. After that, we present the syntax of our calculus, and formally define its operational semantics, by means of a labeled transition system.

Context

A conversation context is a medium where related interactions can take place. A conversation context can be distributed in many pieces, and processes inside any piece can seamlessly talk to any other piece of the same context. Each context has a unique name (cf., a URI), and is partitioned in two endpoints, which we will refer by "initiator" (◀), or "responder" (▶). We use the endpoint access construct

$$n \blacktriangleleft [P]$$

to say that the process P is placed at the initiator endpoint of context n, and the (dual) construct

$$n \blacktriangleright [P]$$

to say that the process P is placed at the responder endpoint of context n. Potentially, each endpoint access will be placed at a different enclosing context. On the other hand, any such endpoint access will necessarily be placed at a single enclosing context. The relationship between the enclosing context and such an endpoint may be seen as a call/callee relationship, but where both entities may interact continuously. Sometimes it is useful to introduce anonymous (or protected) contexts, that may be defined thus (where n is not used in P):

$$[P] \triangleq (\mathtt{new}\, n)(n \blacktriangleright [P])$$

Service Publication and Service Instantiation

A context (a.k.a. a site) may publish one or more service definitions. Service definitions are stateless entities, pretty much as function definitions in a functional programming language. A service definition may be expressed by the construct

$$\mathtt{def}\ \mathit{ServiceName} \Rightarrow \mathit{ServiceBody}$$

where *ServiceName* is the service name, and *ServiceBody* is the process that should be executed at the service endpoint (responder) for each service instance, in other words the service body. In order to be published, such a definition must be inserted into a context, e.g.,

$$\mathit{ServiceProvider} \blacktriangleright [\mathtt{def}\ \mathit{ServiceName} \Rightarrow \mathit{ServiceBody}\ \cdots]$$

Such a published service may be instantiated by means of a instantiation construct

$$\mathtt{instance}\ n\rho\mathit{ServiceName} \Leftarrow \mathit{ClientProtocol}$$

where $n\rho$ describes the context (n) and the endpoint role (ρ) where the service is published. For instance, the service defined above may be instantiated by

$$\texttt{instance}\ \mathit{ServiceProvider} \blacktriangleright \mathit{ServiceName} \Leftarrow \mathit{ClientProtocol}$$

The *ClientProtocol* describes the process that will run inside the initiator endpoint. The outcome of a service instantiation is the creation of a new globally fresh context identity (a hidden name), and the creation of two dual endpoints of a context named by this fresh identity. The responder endpoint will contain the *ServiceBody* process and will be located inside the *ServiceProvider* context. The initiation endpoint will contain the *ClientProtocol* process and will be located in the same context as the **instance** expression that requested the service instantiation.

Context Awareness

A process running inside a given context should be able to dynamically access its identity. This capability may be realized by the construct

$$\texttt{here}(x).P$$

The variable x will be replaced inside the process P by the name n of the current context. The computation will proceed as $P\{x \leftarrow n\}$. This primitive bears some similarity with the **self** or **this** of object-oriented languages, even if it has a different semantics.

Communication

Communication between subsystems is realized by means of message passing. Internal computation is related to communications between subsystems inside the given context. We denote the input and the output of messages from/to the current context by the constructs

$$\texttt{in}\ \downarrow \mathit{Message}(x_1,\ldots,x_n).P$$
$$\texttt{out}\ \downarrow \mathit{Message}(v_1,\ldots,v_n).P$$

In the output case, the terms v_i represent message parameters, values to be sent, as expected. In the input case, the variables x_i represent message parameters and are bound in P, as expected. The target symbol $\downarrow$ (read "here") says that the corresponding communication actions must interact in the current context.

Second, we denote the input and the output of messages from/to the outer context by the constructs

$$\texttt{in}\ \uparrow \mathit{Message}(x_1,\ldots,x_n).P$$
$$\texttt{out}\ \uparrow \mathit{Message}(v_1,\ldots,v_n).P$$

The target symbol $\uparrow$ (read "up") says that the corresponding communication actions must interact in the (uniquely determined) outer context, where "outer" is understood relatively to the context where the **out** $\uparrow$ or **in** $\uparrow$ process is running.

Third, we denote the input and the output of messages from/to the dual endpoint by the constructs

$$\texttt{in}\ \leftarrow \mathit{Message}(x_1,\ldots,x_n).P$$
$$\texttt{out}\ \leftarrow \mathit{Message}(v_1,\ldots,v_n).P$$

The target symbol $\leftarrow$ (read "other") says that the corresponding communication action must interact in the other (the dual) endpoint context, relative to the context where the **out** $\leftarrow$ or **in** $\leftarrow$ process is running.

Exception Handling

We introduce two primitives to model exceptional behavior, in particular fault signaling, fault detection, and resource disposal, these aspects are certainly orthogonal to the previously introduced communication mechanisms. We recover the classical **try** − **catch**− and **throw**−, adapted to a concurrent setting. The primitive to signal exceptional behavior is

$$\textbf{throw}.\mathit{Exception}$$

This construct throws an exception with continuation the process *Exception*, and has the effect of forcing the termination of all other processes running in all enclosing contexts, up to the point where a **try** − **catch** block is found (if any). The continuation *Exception* will be activated when (and if) the exception is caught by such an exception handler. The exception handler construct

$$\textbf{try}\ P\ \textbf{catch}\ \mathit{Handler}$$

actively allows a process P to run until some exception is thrown inside P. At that moment, all of P is terminated, and the *Handler* handler process, which is guarded by **try** − **catch**, is activated, concurrently with the continuation *Exception* of the **throw**.*Exception* that originated the exception, in the context of a given **try** − **catch**− block. By exploiting the interaction potential of the *Handler* and *Exception* processes, one may represent many adequate recovery and resource disposal protocols.

3.1 Syntax and Semantics of the Calculus

We may now formally introduce the syntax and semantics of the conversation calculus. We assume given an infinite set of names Λ, an infinite set of variables $\mathcal{V}$, and an infinite set of labels $\mathcal{L}$. We abbreviate $a_1,\ldots,a_k$ by $\widetilde{a}$. We use *dir* for the communication directions, α for directed message labels, and ρ for the endpoint roles ($\rho = \blacktriangleleft$, the initiator role, or $\rho = \blacktriangleright$, the responder role). We denote by $\overline{\rho}$ the dual role of ρ, for instance $\overline{\blacktriangleright} = \blacktriangleleft$. Notice that message and service identifiers (from $\mathcal{L}$) are plain labels, not subject to restriction or binding. The syntax of the calculus is defined in Figure 3.

The static core of our language is derived from the π-calculus [19]. We thus have **stop** for the inactive process, $P \mid Q$ for the parallel composition, $(\textbf{new}\ a)P$ for name restriction, and $!P$ for replication. Then we have context-oriented polyadic communication primitives: $\textbf{out}\ \alpha(\widetilde{v}).P$ for output and $\textbf{in}\ \alpha(\widetilde{x}).P$ for input. In the communication primitives, α denotes a pair of name and direction, as explained before. We then have the context endpoint access construct $n\,\rho\,[P]$, the context awareness primitive $\textbf{here}(x).P$, the service invocation and service definition primitives $\textbf{instance}\ n\,\rho\,s \Leftarrow P$ and $\textbf{def}\ s \Rightarrow P$, respectively. The primitives for exception handling are the $\textbf{try}\ P\ \textbf{catch}\ Q$ and the $\textbf{throw}.P$. The distinguished occurrences of a, $\widetilde{x}$, and x are binding occurrences in $(\textbf{new}\ a)P$, $\textbf{in}\ \alpha(\widetilde{x}).P$, and $\textbf{here}(x).P$, respectively. The sets of free ($fn(P)$) and bound ($bn(P)$) names and variables in a process P are defined as usual, and we implicitly identify α-equivalent processes.

We define the semantics of the conversation calculus using a labeled transition system. We introduce transition labels λ. We use *act* to range over actions, defined as

$$\mathit{act} \quad ::= \quad \tau \mid \alpha(\widetilde{a}) \mid \texttt{here} \mid \texttt{throw} \mid \texttt{def}\ s$$

Then, a transition label λ is an expression as given by $\lambda ::= c\,\rho\,\mathit{act} \mid \mathit{act} \mid (\nu a)\lambda$. In $(\nu a)\lambda$ the distinguished occurrence of a is bound with scope λ (cf., the π-calculus bound output and bound input actions). A transition label containing $c\,\rho$ is said to be *located at* $c\,\rho$ (or just *located*),

$$
\begin{array}{lcll}
a,b,c,\ldots & \in & \Lambda & \text{(Names)} \\
x,y,z,\ldots & \in & \mathcal{V} & \text{(Variables)} \\
n,v,\ldots & \in & \Lambda \cup \mathcal{V} & \text{(Names and variables)} \\
l,s\ldots & \in & \mathcal{L} & \text{(Labels)} \\
 & & & \\
dir & ::= & \downarrow \mid \leftarrow \mid \uparrow & \text{(Directions)} \\
\alpha & ::= & dir\ l & \text{(Directed message labels)} \\
\rho & ::= & \blacktriangleright \mid \blacktriangleleft & \text{(Roles)}
\end{array}
$$

$$
\begin{array}{lll}
P,Q \ ::= & & \text{(Processes)} \\
 & \texttt{stop} & \text{(Inaction)} \\
\mid & P \mid Q & \text{(Parallel)} \\
\mid & (\texttt{new}\ a)P & \text{(Restriction)} \\
\mid & \texttt{out}\ \alpha(\widetilde{v}).P & \text{(Output)} \\
\mid & \texttt{in}\ \alpha(\widetilde{x}).P & \text{(Input)} \\
\mid & !P & \text{(Replication)} \\
 & & \\
\mid & n\,\rho\,[P] & \text{(Context)} \\
\mid & \texttt{here}(x).P & \text{(Context awareness)} \\
\mid & \texttt{instance}\ n\,\rho\,s \Leftarrow P & \text{(Instantiation)} \\
\mid & \texttt{def}\ s \Rightarrow P & \text{(Definition)} \\
 & & \\
\mid & \texttt{try}\ P\ \texttt{catch}\ Q & \text{(Try-catch)} \\
\mid & \texttt{throw}.P & \text{(Throw)}
\end{array}
$$

Figure 3: The Conversation Calculus

otherwise is said to be *unlocated*. We write $(\widetilde{\nu a})$ to abbreviate a (possibly empty) sequence $(\nu a_1)\ldots(\nu a_k)$.

We adopt a few conventions and notations. We note by λ^{dir} a transition label λ^{dir} containing the direction dir ($\uparrow,\leftarrow,\downarrow$). Then we denote by $\lambda^{dir'}$ the label obtained by replacing dir by dir' in λ^{dir}. Given an unlocated label λ, we represent by $c\,\rho\cdot\lambda$ the label obtained by locating λ at $c\,\rho$, so that e.g., $c\,\rho\cdot(\widetilde{\nu a})act = (\widetilde{\nu a})c\,\rho\,act$. We assert $loc(\lambda)$ if λ is not located and does not contain `here`.

The set of transition labels is polarized and equipped with an injective involution $\overline{\lambda}$ (such that $\overline{\overline{\lambda}} = \lambda$). The involution, used to define synchronizing (matching) transition labels, is defined such that $\overline{act} \neq act'$ for all act, act', and

$$\overline{c\ \rho\ \texttt{def}\ s} \triangleq c\ \rho\ \overline{\texttt{def}\ s} \qquad \overline{c\ \rho\ \downarrow \alpha} \triangleq c\ \rho\ \overline{\downarrow \alpha} \qquad \overline{c\ \rho\ \leftarrow \alpha} \triangleq c\ \overline{\rho}\ \overline{\leftarrow \alpha}$$

We define $out(\lambda)$ as $\widetilde{a}\setminus(\widetilde{b}\cup\{c\})$, if $\lambda = (\widetilde{\nu b})\overline{c\,\rho\,\alpha(\widetilde{a})}$ or $\lambda = (\widetilde{\nu b})\overline{\alpha(\widetilde{a})}$. We use $fn(\lambda)$ and $bn(\lambda)$ to denote (respectively) the free and bound names of a transition label.

In Figs. 4, 5 and 6 we present the labeled transition system for the calculus. The rules presented in Figure 4 closely follow the π-calculus labeled transition system (see [21]). In (vii) the unlocated $\leftarrow$ label is excluded (to synchronize it must first get located in some context). We omit the rule symmetric to (vi).

We briefly review the rules presented in Figure 5: (i) service instantiation request; (ii) service instantiation; (iii) after going through a context boundary, an $\uparrow$ message becomes $\downarrow$; (iv) an

unlocated $\downarrow$ message gets located at the context identity in which it originates, analogously (v) for a $\leftarrow$ message and (vi) for service instantiation; (vii) a `here` label matches the enclosing context; $(viii)$ a `here` label reads the context identity; (ix) a non-`here` located label transparently crosses the context boundary, likewise (x) for a τ label; (xi) an unlocated label synchronizes with a part (the unlocated part) of a located label, originating a `here` label, thus requiring the interaction to occur inside the given context. We omit the rule symmetric to (xi).

As for the rules in Figure 6: (i) signals an exception; (ii) and (iii) terminate enclosing computations, (iv) a non-`throw` transition crosses the handler block, (v) an exception is caught by the handler block. We omit the rule symmetric to (ii).

Notice that the presentation of the transition system is fully modular: the rules for each operator are independent, so that one may easily consider several fragments of the calculus (e.g., without exception handling primitives). The operational semantics of closed systems, usually represented by a reduction relation, is here specified by $\xrightarrow{\tau}$.

4 Examples

In this section, we illustrate the expressiveness of our calculus through a sequence of simple, yet illuminating examples. For the sake of commodity, we informally extend the language with some auxiliary primitives, e.g., $\mathbf{if} - \mathbf{then} - \mathbf{else}$, etc, and standard concurrency combinators, for instance, input guarded choice $\oplus$ (that may be added perhaps to the basic language as a primitive) and recursion $\mathbf{rec}\, X.P$ (that may be represented using replication). We also use anonymous (or protected) contexts, defined as (where n is not used in P) $[P] \triangleq (\mathbf{new}\, n)(n \blacktriangleright [P])$.

4.1 Reading a Remote Value

We start with a trivial example. A service provider *Antarctica* provides a service *Temperature*. Whenever invoked such service reads the current value of a sensor at the service provider site, and sends it to the caller endpoint.

$$
\begin{array}{l}
\mathit{Antarctica} \blacktriangleright [\\
\quad \mathbf{def}\ \mathit{Temperature} \Rightarrow \\
\qquad \mathbf{in} \uparrow \mathit{ReadSensor}(x).\mathbf{out} \leftarrow \mathit{SensorValue}(x) \\
\quad | \\
\quad \mathit{Sensor} \\
]
\end{array}
$$

By *Sensor* we denote some process running in the $\mathit{Antarctica} \blacktriangleright [\cdots]$ context, and that is able to send $\mathit{ReadSensor}(t)$ messages inside that context, where t is the current temperature.

To use the service in "one shot", a remote client may use the code

$$
\begin{array}{l}
\mathbf{instance}\ \mathit{Antarctica} \blacktriangleright \mathit{Temperature} \Leftarrow \\
\qquad \mathbf{in} \leftarrow \mathit{SensorValue}(x).\mathbf{out} \uparrow \mathit{Temperature}(x)
\end{array}
$$

The effect of this code would be to drop a $\mathit{Temperature}(t)$ message in the client context, where t is the temperature value as read by the remote sensor at the *Antarctica* site. A service delegation as the one just shown resembles a remote method call in a distributed object system.

4.2 Memory Cell

We discuss here some simple examples of stateful service definition and invocation patterns, using memory cell implementations. Here is a possible implementation of a memory cell ser-

$$\mathtt{out}\ \alpha(\widetilde{v}).P \xrightarrow{\overline{\alpha(\widetilde{v})}} P \quad (i) \qquad \mathtt{in}\ \alpha(\widetilde{x}).P \xrightarrow{(\widetilde{\nu n})\alpha(\widetilde{v})} P\{\widetilde{x}\leftarrow\widetilde{v}\} \quad (\widetilde{n}\subseteq\widetilde{v}) \quad (ii)$$

$$\frac{P \xrightarrow{\lambda} Q \quad n \notin fn(\lambda)}{(\mathtt{new}\, n)P \xrightarrow{\lambda} (\mathtt{new}\, n)Q}(iii) \qquad \frac{P \xrightarrow{\lambda} Q \quad n \in out(\lambda)}{(\mathtt{new}\, n)P \xrightarrow{(\nu n)\lambda} Q}(iv) \qquad \frac{P \mid !P \xrightarrow{\lambda} Q}{!P \xrightarrow{\lambda} Q}(v)$$

$$\frac{P \xrightarrow{\lambda} Q \quad \lambda \neq \mathtt{throw}}{P \mid R \xrightarrow{\lambda} Q \mid R}(vi) \qquad \frac{P \xrightarrow{(\widetilde{\nu n})\lambda} P' \quad Q \xrightarrow{(\widetilde{\nu n})\overline{\lambda}} Q' \quad \lambda \neq \leftarrow l(\widetilde{a})}{P \mid Q \xrightarrow{\tau} (\mathtt{new}\, \widetilde{n})(P' \mid Q')}(vii)$$

Figure 4: Basic Operators

$$\mathtt{instance}\, n\rho\, s \Leftarrow P \xrightarrow{\overline{(\nu c)n\rho\,\mathtt{def}\, s}} c \blacktriangleleft [P] \quad (i) \qquad \mathtt{def}\, s \Rightarrow P \xrightarrow{(\nu c)\mathtt{def}\, s} c \blacktriangleright [P] \quad (ii)$$

$$\frac{P \xrightarrow{\lambda^{\uparrow}} Q}{n\rho\,[P] \xrightarrow{\lambda^{\downarrow}} n\rho\,[Q]}(iii) \qquad \frac{P \xrightarrow{\lambda^{\downarrow}} Q}{n\rho\,[P] \xrightarrow{n\rho\cdot\lambda^{\downarrow}} n\rho\,[Q]}(iv) \qquad \frac{P \xrightarrow{\lambda^{\leftarrow}} Q}{n\rho\,[P] \xrightarrow{n\rho\cdot\lambda^{\leftarrow}} n\rho\,[Q]}(v)$$

$$\frac{P \xrightarrow{(\nu c)\mathtt{def}\, s} Q}{n\rho\,[P] \xrightarrow{(\nu c)n\rho\,\mathtt{def}\, s} n\rho\,[Q]}(vi) \qquad \frac{P \xrightarrow{n\rho\,\mathtt{here}} Q}{n\rho\,[P] \xrightarrow{\tau} n\rho\,[Q]}(vii) \qquad \mathtt{here}(x).P \xrightarrow{n\rho\,\mathtt{here}} P\{x\leftarrow n\} \quad (viii)$$

$$\frac{P \xrightarrow{\lambda} Q \quad loc(\lambda)}{n\rho\,[P] \xrightarrow{\lambda} n\rho\,[Q]}(ix) \qquad \frac{P \xrightarrow{\tau} Q}{n\rho\,[P] \xrightarrow{\tau} n\rho\,[Q]}(x) \qquad \frac{P \xrightarrow{(\widetilde{\nu n})act} P' \quad Q \xrightarrow{(\widetilde{\nu n})\overline{c\rho\,act}} Q'}{P \mid Q \xrightarrow{c\rho\,\mathtt{here}} (\mathtt{new}\, \widetilde{n})(P' \mid Q')}(xi)$$

Figure 5: Service and Context Operators

$$\mathtt{throw}.P \xrightarrow{\mathtt{throw}} P \quad (i) \qquad \frac{P \xrightarrow{\mathtt{throw}} R}{P \mid Q \xrightarrow{\mathtt{throw}} R}(ii) \qquad \frac{P \xrightarrow{\mathtt{throw}} R}{n\rho\,[P] \xrightarrow{\mathtt{throw}} R}(iii)$$

$$\frac{P \xrightarrow{\lambda} Q \quad \lambda \neq \mathtt{throw}}{\mathtt{try}\, P\, \mathtt{catch}\, R \xrightarrow{\lambda} \mathtt{try}\, Q\, \mathtt{catch}\, R}(iv) \qquad \frac{P \xrightarrow{\mathtt{throw}} R}{\mathtt{try}\, P\, \mathtt{catch}\, Q \xrightarrow{\tau} Q \mid R}(v)$$

Figure 6: Exception Handling Operators

vice.

$$\begin{array}{l}\mathtt{def}\ \mathit{Cell} \Rightarrow (\\ \quad !\mathtt{in} \leftarrow \mathit{Read}().\mathtt{in} \downarrow \mathit{Value}(x).(\mathtt{out} \leftarrow \mathit{Value}(x) \mid \mathtt{out} \downarrow \mathit{Value}(x))\\ \quad \oplus\\ \quad \mathtt{in} \leftarrow \mathit{Write}(x).(\mathtt{out} \downarrow \mathit{Value}(x))\\ )\end{array}$$

We show how to instantiate the *Cell* service so to create a delegate cell process in the current context. The delegate accepts $Put(v)$ and $Get()$ messages from the client context, and replies to each $Get()$ message with a $Reply(v)$ message to the context. It provides the memory cell functionality delegation to the remote service $FreeCellsInc \blacktriangleright Cell$.

$$\begin{array}{l}\mathtt{instance}\ \mathit{FreeCellsInc} \blacktriangleright \mathit{Cell} \Leftarrow (\\ \quad !\mathtt{in} \uparrow \mathit{Put}(x).\mathtt{out} \leftarrow \mathit{Write}(x)\\ \quad \oplus\\ \quad \mathtt{in} \uparrow \mathit{Get}().\mathtt{out} \leftarrow \mathit{Read}().\mathtt{in} \leftarrow \mathit{Value}(x).\mathtt{out} \uparrow \mathit{Reply}(x)\\ )\end{array}$$

A process in the context may then use the created service instance as follows:

$$\mathtt{out} \downarrow \mathit{Put}(\mathit{value}).\mathtt{out} \downarrow \mathit{Get}().\mathtt{in} \downarrow \mathit{Reply}(x).\mathtt{out} \downarrow \mathit{Proceed}(x)$$

4.3 Dictionary

In the next example, we use a toy dictionary service to discuss the possible need of correlating messages belonging to different interaction contexts. A possible instantiation of such a service may be expressed thus:

$$\begin{array}{l}\mathtt{instance}\ \mathit{FreeBagsCo} \blacktriangleright \mathit{Dict} \Leftarrow (\\ \quad !\mathtt{in} \uparrow \mathit{Put}(\mathit{key},x).\mathtt{out} \leftarrow \mathit{Store}(\mathit{key},x)\\ \quad \oplus\\ \quad \mathtt{in} \uparrow \mathit{Get}(\mathit{key}).\mathtt{out} \leftarrow \mathit{Get}(\mathit{key}).\mathtt{in} \leftarrow \mathit{Value}(x).\mathit{Reply}(x)\\ )\end{array}$$

If the generated instance is to be solicited by several concurrent $Get(key)$ requests, some form of correlation may be needed, in order to route the associated $Reply(v)$ answers to the appropriate contexts. In this case, we set the $Get(r,key)$ message to play the role of an initiator message, now receiving also a reference r to the context of interaction (associated to getting the dictionary entry associated to the key *key*).

$$\begin{array}{l}\mathtt{instance}\ \mathit{FreeBagsCo} \blacktriangleright \mathit{Dict} \Leftarrow (\\ \quad !\mathtt{in} \uparrow \mathit{Put}(\mathit{key},x).\mathtt{out} \leftarrow \mathit{Store}(\mathit{key},x)\\ \quad \oplus\\ \quad \mathtt{in} \uparrow \mathit{Get}(r,\mathit{key}).\mathtt{out} \leftarrow \mathit{Get}(\mathit{key}).\mathtt{in} \leftarrow \mathit{Value}(x).r \blacktriangleright [\mathtt{out} \downarrow \mathit{Reply}(x)]\\ )\end{array}$$

Now, the $Reply(v)$ message is sent inside the appropriate conversation context r, the one that relates to the initial $Get(r)$. A process in the context may then use the service instance by following the appropriate intended protocol, e.g.:

$$\begin{array}{l}\mathtt{out} \downarrow \mathit{Put}(\mathit{key},\mathit{value}).\\ (\mathtt{new}\ r)(\\ \quad \mathtt{out} \downarrow \mathit{Get}(r,\mathit{key}).\\ \quad r \blacktriangleright [\mathtt{in} \downarrow \mathit{Reply}(x).\mathtt{out} \uparrow \mathit{Proceed}(x)])\end{array}$$

Here, we are essentially in presence of a familiar form of continuation passing. We present later another example on the correlation of interactions, in a sense closer to the one used in web services technology.

In this case, we have generated a new special context r in order to carry out the appropriate conversation. In many situations we would like just to correlate the subsidiary conversation with the current context, without having to introduce a new special context. In this case, we may write the (perhaps more natural) code, that will have the same effect as the code above:

$$\begin{array}{l}
\mathtt{out} \downarrow \mathit{Put}(\mathit{key}, \mathit{value}). \\
\mathtt{here}(\mathit{thiscontext}). \\
\mathtt{out} \downarrow \mathit{Get}(\mathit{thiscontext}, \mathit{key}). \\
\mathtt{in} \downarrow \mathit{Reply}(x). \\
\mathtt{out} \uparrow \mathit{Proceed}(x)
\end{array}$$

Remember that the $\mathtt{here}(r).P$ (context-awareness) primitive binds r in P to the identity of the current context.

4.4 Service Provider Factory

We revisit the memory cell example, and provide a different realization. In this case, we would like to represent each cell as a specific service provider, such that the *Read* and *Write* operations are now services, rather than operations of a particular service as shown above. A cell (named n) may be be represented by the context:

$$\begin{array}{rcll}
\mathit{Cell}(n) & \triangleq & n \blacktriangleright [& \mathtt{def}\ \mathit{Read} \Rightarrow \mathtt{in} \uparrow \mathit{Value}(x).\mathtt{out} \leftarrow \mathit{Value}(x) \mid \\
& & & \mathtt{def}\ \mathit{Write} \Rightarrow \mathtt{in} \leftarrow \mathit{Value}(x).\mathtt{out} \uparrow \mathit{Value}(x) \quad]
\end{array}$$

We may now specify a memory cell factory service.

$$\mathit{CellFactoryService} \triangleq \mathtt{def}\ \mathit{NewCell} \Rightarrow (\mathtt{new}\ n)(\mathit{Cell}(n) \mid \mathtt{out} \leftarrow \mathit{ReplyCell}(n))$$

To instantiate the cell factory service, and drop a $\mathit{TheCell}(c)$ message with a fresh cell reference (c) in the current context, we may write:

$$\mathtt{instance}\ \mathit{FreeCellsInc} \blacktriangleright \mathit{NewCell} \Leftarrow \mathtt{in} \leftarrow \mathit{ReplyCell}(x).\mathtt{out} \uparrow \mathit{TheCell}(x)$$

The newly allocated cell service provider is allocated in the *FreeCellsInc* context, as expected. To use the cell one may then write, e.g.,

$$\begin{array}{l}
\mathtt{in} \downarrow \mathit{TheCell}(c).(\\
\cdots \\
\mathtt{instance}\ c \blacktriangleright \mathit{Read} \Leftarrow \cdots \\
\mid \cdots \\
\mathtt{instance}\ c \blacktriangleright \mathit{Write} \Leftarrow \cdots \\
\cdots \\
)
\end{array}$$

This usage pattern for services, where service instantiation corresponds to some form of task delegation rather that process delegation, is closer to a distributed object model than to a service-oriented model. In any case, it is interesting to be able to accommodate this usage pattern as a special case, not only for the sake of abstract generality, but also because it will certainly turn out useful in appropriate scenarios.

```
def TravelApproval ⇒
  in ← TravelRequest(employee,flightData).
  out ↑ EmployeeTravelStatusRequest(employee).
  in ↑ EmployeeTravelStatusResponse(travelClass).
   (
     instance AmericanAirlines ▶ FlightAvailability ⇐
       out ← FlightDetails(flightData,travelClass).
       in ← FlightTicketCallBack(responseAA,priceAA).
       out ↑ FlightResponseAA(responseAA,priceAA).
       out ← Done()
       |
     instance DeltaAirlines ▶ FlightAvailability ⇐
       out ← FlightDetails(flightData,travelClass).
       in ← FlightTicketCallBack(responseDA,priceDA).
       out ↑ FlightResponseDA(responseDA,priceDA).
       out ← Done()
       |
       in ↓ FlightResponseAA(flightAA,priceAA).
       in ↓ FlightResponseDA(flightDA,priceDA).
       if (priceAA < priceDA) then
           out ← ClientCallBack(flightAA)
         else
           out ← ClientCallBack(flightDA)
   )
```

Figure 7: The Travel Approval Service (I).

4.5 Service Composition and Orchestration (I)

Our next example, depicted in in Figure 7, illustrates a familiar service composition and orchestration scenario (inspired by a tutorial example on BPEL published in the Oracle website [15]). Any instance of the *TravelApproval* service is expected to receive a TravelRequest message and return a *ClientCallBack* message after finding a suitable flight. The implementation of the service relies on subsidiary services provided by *AmericanAirlines* and *DeltaAirlines* in order to identify the most favorable price.

Notice how the service instance interacts with service side resources in order to find the *travelClass* associated to each *employee*, by means of the *EmployeeTravelStatusRequest* and *EmployeeTravelStatusResponse* messages to and from the server endpoint context. The context is also used to pass around control messages with the responses from the two airline services involved, *FlightResponseAA* and *FlightResponseDA*, respectively.

In the `WSBPEL` example cited above, information about the flights is stored in state variables of the script, manipulated by `WSBPEL` "assign" commands. Such contextual state manipulations are represented in our model by interactions between the endpoints and their context via upward (↑) message exchanges. Another difference between the above specification and the `WSBPEL` code is the separation between service instantiation and (initiating) message exchange, an unfortunate design decision of `WSBPEL` to mix up these two concepts. Apart from these superficial differences, we believe that the code above is a faithful and succinct rendering of the inspiring `WSBPEL` script.

```
def TravelApproval ⇒ (
   instance AmericanAirlines ▶ FlightAvailability ⇐      % Partner AmericanAirlines
      in ↑ FlightRequestAA(flightData, travelClass).
      out ← FlightDetails(flightData, travelClass).
      in ← FlightTicketCallBack(response, price).
      out ↑ FlightResponseAA(response, price).out ← Done()
   |
   instance DeltaAirlines ▶ FlightAvailability ⇐          % Partner DeltaAirlines
      in ↑ FlightRequestDA(flightData, travelClass).
      out ← FlightDetails(flightData, travelClass).
      in ← FlightTicketCallBack(response, price).
      out ↑ FlightResponseDA(response, price).out ← Done()
   |
   in ← TravelRequest(employee, flightData).              % Orchestration
   out ↑ EmployeeTravelStatusRequest(employee).
   in ↑ EmployeeTravelStatusResponse(travelClass).(
      out ↓ FlightRequestAA(flightData, travelClass) |
      out ↓ FlightRequestDA(flightData, travelClass))
   |
   in ↓ FlightResponseAA(flightAA, priceAA).
   in ↓ FlightResponseDA(flightDA, priceDA).
   if (priceAA < priceDA) then
      out ← ClientCallBack(flightAA)
   else
      out ← ClientCallBack(flightDA)
)
```

Figure 8: The Travel Approval Service (II)

4.6 Service Composition and Orchestration (II)

More substantial differences on the structure of the BPEL script and our rendering above relates to the declaration of the so-called partner links and partner roles. These are BPEL concepts introduced with the motivation of decoupling the description of the business process (the workflow) from the identification and binding to the actual partners involved in the particular service instances. We present in Figure 8 a different rendering of the BPEL script under consideration, based on our view of service invocation as process delegation, that assigns some meaning to the intuitive notions of partner links and roles. The separation between the partner service instances and the orchestration script is made clearer in this presentation.

All interactions between the orchestration and the endpoints is loosely-coupled and realized through messages exchanged in the context of each particular *TravelApproval* service instance, but also with the external context. The body of this service definition follows the general pattern

$$\begin{array}{l}\textbf{instance}\ Partner_1 \blacktriangleright Service_1 \Leftarrow Wire_1 \mid \\ \ldots \\ \textbf{instance}\ Partner_n \blacktriangleright Service_n \Leftarrow Wire_n \mid \\ OrchestrationProcess\end{array}$$

where *OrchestrationProcess* is a process communicating with the several instance via mes-

sages, and the $Wire_i$ descriptions adapt the remote endpoint functionalities (or protocols) to the particular roles performed by the instances in this local process.

4.7 Service Composition and Orchestration (III)

In Figure 9 we elaborate a bit on the last example, with the intent of allowing the client of the service to obtain several possible travel plans for the *TravelRequest* before committing. This will require unbounded message exchanges (conversations) between the "orchestration" process and the (two in this case) subsidiary service instances. In this case, upon reception of the $ClientCallBack(flight)$ message the client endpoint is expected to either finalize the instance by means of a $Done()$ message, or to ask for another flight, by means of a $Retry()$ message.

4.8 Logging flight data by delegating to Amazon S3

We further extend our previous example, in Figure 10, by adding a delegate to (a service inspired in) Amazon S3 Simple Storage Service [13], in order to store the flight data processed. To do that we just introduce a new "partner", and modify the workflow just with a message sending operation $LogFlight(flightData)$ (marked below). Notice how the modifications to the orchestration script are quite independent of the particular storage provider chosen.

4.9 Service Composition and Orchestration (IV)

We get back to example 4.7, to discuss another interesting variation. In this case, we would like to instantiate the *FlightAvailability* services independently (e.g., at site setup time), in the service provider context, rather than creating new instances for each instantiation of the *TravelApproval* service. In other words, the service *DeltaAirlines* ▶ *FlightAvailability* and the service *AmericanAirlines* ▶ *FlightAvailability* will be used by the orchestration script in the same way as the $EmployeeTravelStatus$ already is, by means of loosely coupled message exchanges. We depict the solution in Figure 11.

Since many concurrent instantiations of the *TravelApproval* service may be outstanding at any given moment, the need arises to explicitly keep track of the messages relative to each instance (establish a correlation mechanism, in web services technology terminology). In the specification above, correlation is achieved by passing the name of the current endpoint context (accessed by the `here`$(context)$ primitive) in the appropriate messages to the services instantiated in the shared context (e.g., as in the message $FlightRequestAA(context, flightData)$).

4.10 Orc

The Orc language by Misra and Cook [20] is sometimes cited as presenting a general model of service orchestration. This example is of particular interest to our discussion, since Orc also seems to present a mechanism of process delegation, although in a more restricted sense than we are introducing here. In fact, calling a site in Orc causes a persistent process to be spawned, the observable behavior of such a process consists in streaming a sequence of values to athe

```
def TravelApproval ⇒
  instance AmericanAirlines ▶ FlightAvailability ⇐
    rec Loop.
      in ↑ FlightRequestAA(flightData, travelClass).
      out ← FlightDetails(flightData, travelClass).
      in ← FlightTicketCallBack(response, price).
      out ↑ FlightResponseAA(response, price).Loop
      ⊕
      in ↑ DoneAA().out ← Done()
  |
  instance DeltaAirlines ▶ FlightAvailability ⇐
    rec Loop.
      in ↑ FlightRequestDA(flightData, travelClass).
      out ← FlightDetails(flightData, travelClass).
      in ← FlightTicketCallBack(response, price).
      out ↑ FlightResponseDA(response, price).Loop
      ⊕
      in ↑ DoneDA().out ← Done()
  |
  in ← TravelRequest(employee, flightData).
  out ↑ EmployeeTravelStatusRequest(employee).
  in ↑ EmployeeTravelStatusResponse(travelClass).
  rec Start.
    (
        out ↓ FlightRequestAA(flightData, travelClass) |
        out ↓ FlightRequestDA(flightData, travelClass) |
        in ↓ FlightResponseAA(flightAA, priceAA).
        in ↓ FlightResponseDA(flightDA, priceDA).
        if (priceAA < priceDA) then
            out ↓ Prompt(flightAA)
          else
            out ↓ Prompt(flightDA)
        |
        in ↓ Prompt(flight).
        out ← ClientCallBack(flight).
          in ← Done().(out ↑ DoneAA() | out ↑ DoneDA())
          ⊕
          in ← Retry().Start
    )
```

Figure 9: Service Travel Approval (III)

```
def TravelApproval ⇒
  instance Amazon ▶ S3 ⇐                          % Added instance of Amazon.S3
    out ← Authenticate(credentials).
    in ← AccessKey(SessionAccessKey).
    rec Loop. in ↑ LogFlight(flightData).
      out ← PutObject(SessionAccessKey, travelClassbucket, flightData).Loop
  |
  instance AmericanAirlines ▶ FlightAvailability ⇐
    ...                                                          % As before
  |
  instance DeltaAirlines ▶ FlightAvailability ⇐
    ...                                                          % As before
  |
  in ← TravelRequest(employee, flightData).
  out ↑ EmployeeTravelStatusRequest(employee).
  in ↑ EmployeeTravelStatusResponse(travelClass).
  rec Start.
    (
        out ↓ FlightRequestAA(flightAA, travelClass) |
        out ↓ FlightRequestDA(flightDA, travelClass) |
        in ↓ FlightResponseAA(flightAA, priceAA).
        in ↓ FlightResponseDA(flightDA, priceDA).
        if (priceAA < priceDA) then
            out ↓ Prompt(flightAA)
          else
            out ↓ Prompt(flightDA)
        |
        in ↓ Prompt(flight).
        out ← ClientCallBack(flight).
        in ↓ LogFlight(flight).                                  % Line inserted
          in ← Done().(out ↑ DoneAA() | out ↑ DoneDA())
          ⊕
          in ← Retry().Start
    )
```

Figure 10: Logging flight data to Amazon S3.

```
instance AmericanAirlines ▶ FlightAvailability ⇐
      ! in ↑ FlightRequestAA(r,flightData,travelClass).
        out ← FlightDetails(flightData,travelClass).
        in ← FlightTicketCallBack(responseAA,priceAA).
        r ▶ [out ↓ FlightResponseAA(responseAA,priceAA)]
|
instance DeltaAirlines ▶ FlightAvailability ⇐
      ! in ↑ FlightRequestDA(r,flightData,travelClass).
        out ← FlightDetails(flightData,travelClass).
        in ← FlightTicketCallBack(responseDA,priceDA).
        r ▶ [out ↓ FlightResponseDA(responseDA,priceDA)]
|
! def TravelApproval ⇒
   in ← TravelRequest(employee,flightData).
   here(context).
   out ↑ EmployeeTravelStatusRequest(context,employee).
   in ↓ EmployeeTravelStatusResponse(travelClass).
   rec Start.
      (
      out ↑ FlightRequestAA(context,flightData,travelClass) |
      out ↑ FlightRequestDA(context,flightData,travelClass) |
      in ↓ FlightResponseAA(flightAA,priceAA).
      in ↓ FlightResponseDA(flightDA,priceDA).
      if (priceAA < priceDA) then
         out ↓ Prompt(flightAA)
            else
         out ↓ Prompt(flightDA)
      |
      in ↓ Prompt(flight).
         out ← ClientCallBack(flight).
            in ← Done()
            ⊕
            in ← Retry().Start
      )
```

Figure 11: Correlating Concurrent Conversations.

caller context.

$$
\begin{array}{lcl}
[\![n.S(x)]\!]_{out} & \triangleq & \mathtt{instance}\ n \blacktriangleright S \Leftarrow \\
 & & \quad (\mathtt{out} \leftarrow args(x).!\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)) \\
[\![n.S(x) = e]\!] & \triangleq & n \blacktriangleright [\mathtt{def}\ S \Rightarrow \\
 & & \quad (\mathtt{in} \leftarrow args(x).[\![e]\!]_{out} \mid \\
 & & \quad !\mathtt{in} \downarrow out(x).\mathtt{out} \leftarrow result(x))] \\
[\![f \gg x \gg g]\!]_{out} & \triangleq & [[\![f]\!]_{out_1} \mid \\
 & & \quad !\mathtt{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x))] \\
[\![f\ \mathtt{where}\ x :\in g]\!]_{out} & \triangleq & [(\mathtt{new}\ x)(\\
 & & \quad [\![f]\!]_{out} \mid \\
 & & \quad !\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid \\
 & & \quad \mathtt{try} \\
 & & \quad\quad [\![g]\!]_{out_2} \mid \mathtt{in} \downarrow out_2(y).\mathtt{throw}\ x \blacktriangleright [!\,\mathtt{out} \leftarrow val(y)] \\
 & & \quad \mathtt{catch}\ \mathbf{0})] \\
[\![x]\!]_{out} & \triangleq & x \blacktriangleleft [\mathtt{in} \leftarrow val(y).\mathtt{out} \uparrow out(y)] \\
[\![f \mid g]\!]_{out} & \triangleq & [\![f]\!]_{out} \mid [\![g]\!]_{out} \\
[\![\mathbf{0}]\!]_{out} & \triangleq & \mathbf{0}
\end{array}
$$

We denote by $[\![O]\!]_{out}$ the encoding of an Orc process O into a conversation calculus process. The *out* parameter identifies the message label used to output the stream of values generated by the Orc process. We consider that Orc's site calls refer the name of the site, and use site calls for both expressions and primitive site calls. The former are located on some given name and the latter on site p. Since primitive site calls can only return a value one could consider a separate case of the encoding for primitive site calls (by simply removing the replication).

$$[\![p.S(x)]\!]_{out} \triangleq \mathtt{instance}\ p \blacktriangleright S \Leftarrow (\mathtt{out} \leftarrow args(x).\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x))$$

In accordance with the Orc semantics [16] we consider that primitive site calls interact with the external environment. We can also encode the Orc's processes resulting from interactions with the external environment as follows

$$
\begin{array}{lcl}
[\![?u]\!] & \triangleq & u \blacktriangleleft [\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)] \\
[\![let(c)]\!] & \triangleq & \mathtt{out} \downarrow out(c)
\end{array}
$$

As for expression definitions we encode them as definitions available on sites that will be available internally to the system, hence we consider an Orc encoding to hold not only the encoding of the Orc process itself but also the encoding of the expression definitions it uses.

To simplify presentation in the encoding of the **where** construct we assume that occurrences of x in site calls $n.S(x)$ in f are replaced by

$$[[\![x]\!]_{out} \mid \mathtt{in} \downarrow out_1(x).[\![n.S(x)]\!]_{out_2} \mid \mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)]$$

We establish the following correspondence between Orc transition labels and Conversation calculus transition labels, and denote by $match_{out}$ the function from the first to (sequences of) the second with an auxiliar parameter *out*. To simplify presentation we abbreviate $\overset{\overline{(\nu u)n \blacktriangleright def\ S}}{\Rightarrow}$ $\overset{\overline{u \blacktriangleleft \leftarrow args(c)}}{\Rightarrow}$ with a single transition where the label results from the sequencing of the two labels $\overset{(\overline{(\nu u)n \blacktriangleright def\ S}, \overline{u \blacktriangleleft \leftarrow args(c)})}{\Rightarrow}$.

$$
\begin{array}{lcl}
match_{out}(!c) & \triangleq & \overline{\downarrow out(c)} \\
match_{out}(\tau) & \triangleq & \tau \\
match_{out}(n.S < c, u >) & \triangleq & (\overline{(\nu u)n \blacktriangleright def\ S}, \overline{u \blacktriangleleft \leftarrow args(c)}) \\
match_{out}(u?c) & \triangleq & u \blacktriangleleft \leftarrow result(c)
\end{array}
$$

We establish an operational correspondence property between this encoding and the formal semantics presented in [16].

Proposition 4.1 (Operational correspondence) *Let O be an Orc process and D the set of expression definitions it uses and $\widetilde{n}$ the set of the names of the contexts where these expression definitions are located. We have that any sequence of transitions O performs can be mimicked by a sequence of matching transitions of $(\mathtt{new}\ \widetilde{n})(\llbracket O \rrbracket_{out} \mid \llbracket D \rrbracket)$ and conversely. We abbreviate $(\mathtt{new}\ \widetilde{n})(\llbracket O \rrbracket_{out} \mid \llbracket D \rrbracket)$ with P_0 and thus write*

$$\begin{array}{ccccccccccc} O & \xrightarrow{l_1} & O_1 & \xrightarrow{l_2} & O_2 & \xrightarrow{l_3} & \dots & \xrightarrow{l_k} & O_k & \xrightarrow{l_{k+1}} & \dots \\ \Longleftrightarrow & & & & & & & & & & \\ P_0 & \overset{match_{out}(l_1)}{\Rightarrow} & P_1 & \overset{match_{out}(l_2)}{\Rightarrow} & P_2 & \overset{match_{out}(l_3)}{\Rightarrow} & \dots & \overset{match_{out}(l_k)}{\Rightarrow} & P_k & \overset{match_{out}(l_{k+1})}{\Rightarrow} & \dots \end{array}$$

4.11 Distributed Objects

Distributed objects are service based systems where the delegated tasks are simple actions, e.g., processes that are known to terminate after performing some work, corresponding to the object's methods. In [6] we have introduced a distributed object calculus in order to study a type system for concurrency control based on spatial logic, please see the above reference for a detailed description of the calculus syntax and semantics. We show how such a model may be represented using our primitives, as follows.

$$\begin{array}{lcl}
\llbracket n.\mathtt{l}(v) \rrbracket_{ok} & \triangleq & \mathtt{instance}\ n \blacktriangleright \mathtt{l} \Leftarrow \\
 & & \quad (\mathtt{out} \leftarrow Args(v).\mathtt{in} \leftarrow Result(x).\mathtt{out} \uparrow ok(x)) \\
\llbracket a? \rrbracket_{ok} & \triangleq & \mathtt{instance}\ a \blacktriangleright Read \Leftarrow \mathtt{in} \leftarrow Value(x).\mathtt{out} \uparrow ok(x) \\
\llbracket a!(v) \rrbracket_{ok} & \triangleq & \mathtt{instance}\ a \blacktriangleright Write \Leftarrow \mathtt{out} \leftarrow Value(v).\mathtt{out} \uparrow ok(\star) \\
\llbracket \mathtt{let}\ x_i = e_i\ \mathtt{in}\ f \rrbracket_{ok} & \triangleq & [\ \llbracket e_1 \rrbracket_{ok_1} \mid \cdots \mid \llbracket e_n \rrbracket_{ok_n} \mid \\
 & & \quad \mathtt{in} \downarrow ok_1(x_1). \cdots .\mathtt{in} \downarrow ok_n(x_n).\llbracket f \rrbracket_{ok} \mid \\
 & & \quad \mathtt{in} \downarrow ok(x).\mathtt{out} \uparrow ok(x)\ ] \\
\llbracket \mathtt{new}\ \overline{[M]} \rrbracket_{ok} & \triangleq & (\mathtt{new}\ n)(n \blacktriangleright [\llbracket M \rrbracket \mid \mathtt{out} \uparrow ok(n)] \\
\llbracket \mathtt{l}(x) = e \rrbracket & \triangleq & \mathtt{def}\ \mathtt{l} \Rightarrow (\mathtt{in} \leftarrow Args(x).\llbracket e \rrbracket_{ok} \mid \mathtt{in} \downarrow ok(x).\mathtt{out} \leftarrow Result(x))
\end{array}$$

We denote by $\llbracket D \rrbracket_{ok}$ the encoding of distribute object calculus process D into a conversation calculus process. The *ok* parameter identifies the message label used to output the results of the expressions.

4.12 Exceptions

We illustrate a few usage idioms for our exception handling primitives. In the first example, the service *Service* is instantiated on site *Server*, and repeatedly re-launched on each failure – failure will be signaled by exception throwing within the local protocol *ClientProto*, possibly as a result of a remote message.

```
rec Restart.
  try
    instance Server ▶ Service ⇐ ClientProto
  catch Restart
```

A possible scenario of remote exception throwing is illustrated below.

$$
\begin{array}{l}
\mathit{Server} \blacktriangleright [\\
\quad \mathbf{def}\ \mathit{Interruptible} \Rightarrow \\
\quad\quad \mathbf{in} \downarrow \mathit{StopRequest}().\mathbf{out} \leftarrow \mathit{UrgentStop}().\mathbf{throw} \mid \\
\quad\quad \ldots \mathit{ServiceProto} \ldots] \\
\\
\mathbf{instance}\ \mathit{Server} \blacktriangleright \mathit{Interruptible} \Leftarrow \\
\quad \mathbf{in} \leftarrow \mathit{UrgentStop}().\mathbf{throw} \mid \\
\quad \ldots \mathit{ClientProto} \ldots
\end{array}
$$

Here, any remote endpoint instance of the *Interruptible* service may be interrupted by the service protocol *ServiceProto* by dropping a message *StopRequest* inside the endpoint context. In this example, such a message causes the endpoint to send an *UrgentStop* message to the other (client side) endpoint, and then throwing an exception, which will cause abortion of the service endpoint. On the other hand, the service invocation protocol will throw an exception at the client endpoint upon reception of *UrgentStop*. Notice that this behavior will happen concurrently with ongoing interactions between *ServiceProto* and *ClientProto*. In this example, the exception issued in the server and client endpoints will have to be managed by appropriate handlers in both sites. In the next example, no exception will be propagated to the service site, but only to the remote client endpoint, and as a result of any exception thrown in the service protocol *ServiceProto*.

$$
\begin{array}{l}
\mathit{Server} \blacktriangleright [\\
\quad \mathbf{def}\ \mathit{Interruptible} \Rightarrow \\
\quad\quad \mathbf{try} \\
\quad\quad\quad \mathit{ServiceProto} \\
\quad\quad \mathbf{catch}\ \mathbf{out} \leftarrow \mathit{UrgentStop}().\mathbf{throw}]
\end{array}
$$

In the examples discussed above, the decision to terminate the ongoing remote interactions is triggered by the service code. In the next example, we show a simple variation of the idioms above, where the decision to kill the ongoing service instance is responsibility of the service context. Here, any instance of the *Interruptible* service may be terminated by the service provider by means of dropping a message *KillRequest* in the endpoint external context.

$$
\begin{array}{l}
\mathit{Server} \blacktriangleright [\\
\quad \mathbf{def}\ \mathit{Interruptible} \Rightarrow \\
\quad\quad \mathbf{try} \\
\quad\quad\quad \mathbf{in} \uparrow \mathit{KillRequest}().\mathbf{throw} \mid \mathit{ServiceProto} \\
\quad\quad \mathbf{catch}\ \mathbf{out} \leftarrow \mathit{UrgentStop}().\mathbf{throw}]
\end{array}
$$

A simple example of a similar pattern in our last example on exceptions.

$$
\begin{array}{l}
\mathit{Server} \blacktriangleright [\\
\quad \mathbf{def}\ \mathit{TimeBound} \Rightarrow \\
\quad\quad \mathbf{in} \uparrow \mathit{TimeAllowed}(\mathit{delay}).\mathbf{wait}(\mathit{delay}).\mathbf{throw} \mid \\
\quad\quad \mathit{ServiceProto}]
\end{array}
$$

Here, any invocation of the *TimeBound* service will be allocated no more than *delay* time units before being interrupted, where *delay* is a dynamic parameter value read from the current server side context (we assume some extension of our sample language with a $\mathbf{wait}(t)$ primitive, with the obvious intuitive semantics).

4.13 Compensations

Although well known in the context of transaction processing systems for quite a long time (see e.g., [10]), the use of compensation as a mechanism to undo the effect of long running transactions, and thus recover some properties of confined ACID transactions (at least consistency and durability), is now frequently assumed to be the recovery mechanism of choice for aborted transactions in distributed services. In this context, some confusion sometimes arises between the notions of exception and compensation: obviously these are quite different and even independent concepts. Exceptions are a mechanism to signal abnormal conditions during program execution, while compensations are commands intended to undo the effects of previously successfully completed tasks during a transaction. Sometimes, an exception mechanism may be a useful tool to describe a compensation mechanism, but this does not need to be always the case. Clearly, compensations are most useful as a structuring device to describe undoable actions, and makes particular sense when the underlying process language is based on a concept of primitive action. Given this understanding, it is not difficult to express a structured compensation mechanism using our primitives.

We illustrate a possible approach by encoding the core fragment of the Compensating CSP calculus presented in [5]. The starting point is the notion of basic action, here we consider a basic action to be any process P, Q that, after successful completion, sends (only once) the message ok to its environment. A basic action is always supposed to enjoy the following property: it either executes successfully to completion, or it aborts. It the case of abortion, a basic action is required not to perform any interesting relevant actions, except signaling abortion by throwing an exception. Structured transactions are then defined from basic compensatable transactions by composition under sequential and parallel composition operations. A basic compensatable transaction is a pair $P\%Q$ where P and Q are basic actions. The intuition is that the basic action Q is able to undo the effect of the P basic action, leading to a state that should be in some sense equivalent to the state right before P was executed. By composing basic actions as explained, one may then obtain a transaction T. A transaction T may then be encapsulated as a basic action, by means of the operator $\langle T \rangle$, enjoying the fundamental properties of a basic action described above.

We may then present our encoding as show in Figure 12. We denote by $[\![P]\!]_{ok}$ the encoding of basic actions, including closed transactions, represented by P, into a conversation calculus process. The ok index represents the message label that signals the successful completion of the basic action, while abortion of a basic action is signaled by throwing an exception. We denote by $[\![PP]\!]_{ok,ab,cm,cb}$ the encoding of a compensatable transaction. We use ok to signal successful completion and ab to signal abortion. With respect to compensation activation, cm is used to trigger the current compensation, and cb to trigger compensations of previously completed actions. A correctness proof of the encoding requires, we believe, a suitable characterization of the effect of compensations.

Notice that we left open the encoding of basic actions other than closed transactions. However, it is not difficult to imagine how to represent a remote task invocation primitive, e.g.,

$$[\![n\rho s]\!]_{ok} \quad \triangleq \quad \texttt{instance}\ n\rho s \Leftarrow (\texttt{out} \leftarrow ok.\texttt{out} \uparrow ok \mid \texttt{out} \leftarrow ko.\texttt{throw})$$

A matching remote service might then be defined by

$$\texttt{def}\ s \Rightarrow (\texttt{try}\ (P \mid \texttt{in} \downarrow ok.\texttt{out} \leftarrow ok)\ \texttt{catch}\ \texttt{out} \leftarrow ko.\texttt{throw})$$

where for the service code P we may consider any basic action (including a compensating transaction). In this way we expect it to be rather straightforward to model general distributed nested compensating transactions.

$$
\begin{array}{lcl}
[\![P\%Q]\!]_{ok,ab,cm,cb} & \triangleq & [\mathtt{try}\ [\![P]\!]_{ok}\ \mathtt{catch}\ (\mathtt{out} \uparrow ab.\mathtt{out} \uparrow cb)\ | \\
 & & \mathtt{in} \downarrow ok.\mathtt{out} \uparrow ok.(\mathtt{in} \uparrow cm.[\![Q]\!]_{cb}\ |\ \mathtt{in} \downarrow cb.\mathtt{out} \uparrow cb)] \\
[\![T_1;T_2]\!]_{ok,ab,cm,cb} & \triangleq & [[\![T_1]\!]_{ok_1,ab_1,cm_1,cb}\ |\ \mathtt{in} \downarrow ab_1.\mathtt{out} \uparrow ab.\mathtt{in} \downarrow cb.\mathtt{out} \uparrow cb\ | \\
 & & \mathtt{in} \downarrow ok_1.[\![T_2]\!]_{ok,ab,cm,cm_1}\ | \\
 & & \mathtt{in} \downarrow ab.\mathtt{out} \uparrow ab.\mathtt{in} \downarrow cb.\mathtt{out} \uparrow cb\ | \\
 & & \mathtt{in} \downarrow ok.\mathtt{out} \uparrow ok.\mathtt{in} \uparrow cm.\mathtt{out} \downarrow cm] \\
[\![T_1 \mid T_2]\!]_{ok,ab,cm,cb} & \triangleq & [\mathtt{in} \downarrow cb_1.\mathtt{in} \downarrow cb_2.\mathtt{out} \uparrow cb\ | \\
 & & [\![T_1]\!]_{ok_1,ab,cm_1,cb_1}\ | \\
 & & [\![T_2]\!]_{ok_2,ab,cm_2,cb_2}\ | \\
 & & \mathtt{in} \downarrow ok_1.\mathtt{in} \downarrow ok_2.\mathtt{out} \uparrow ok. \\
 & & \quad \mathtt{in} \uparrow cm.(\mathtt{out} \downarrow cm_1\ |\ \mathtt{out} \downarrow cm_2)\ | \\
 & & \mathtt{in} \downarrow ab.\mathtt{out} \uparrow ab. \\
 & & \quad (\mathtt{in} \downarrow ab\ |\ \mathtt{in} \downarrow ok_1.\mathtt{out} \downarrow cm_1\ |\ \mathtt{in} \downarrow ok_2.\mathtt{out} \downarrow cm_2)] \\
[\![\mathtt{throww}]\!]_{ok,ab,cm,cb} & \triangleq & [\mathtt{out} \uparrow ab.\mathtt{out} \uparrow cb] \\
[\![\mathtt{skipp}]\!]_{ok,ab,cm,cb} & \triangleq & [\mathtt{out} \uparrow ok.\mathtt{in} \uparrow cm.\mathtt{out} \uparrow cb] \\
[\![\langle T \rangle]\!]_{ok} & \triangleq & [[\![T]\!]_{ok,ab,cm,cb}\ |\ \mathtt{in} \downarrow ab.\mathtt{in} \downarrow cb.\mathtt{throw}\ |\ \mathtt{in} \downarrow ok.\mathtt{out} \uparrow ok] \\
[\![\mathtt{try}\ T_1\ \mathtt{catch}\ T_2]\!]_{ok,ab,cm,cb}{}^{1} & \triangleq & [[\![T_1]\!]_{ok_1,ab_1,cm_1,cb_1} \\
 & & \mathtt{in} \downarrow ok_1.\mathtt{out} \uparrow ok. \\
 & & \quad \mathtt{in} \uparrow cm.(\mathtt{out} \downarrow cm_1\ |\ \mathtt{in} \downarrow cb_1.\mathtt{out} \uparrow cb)\ | \\
 & & \mathtt{in} \downarrow ab_1.\mathtt{in} \downarrow cb_1. \\
 & & \quad ([\![T_2]\!]_{ok_2,ab_2,cm_2,cb_2}\ | \\
 & & \quad \mathtt{in} \downarrow ok_2.\mathtt{out} \uparrow ok. \\
 & & \quad\quad \mathtt{in} \uparrow cm.(\mathtt{out} \downarrow cm_2\ |\ \mathtt{in} \downarrow cb_2.\mathtt{out} \uparrow cb)\ | \\
 & & \quad \mathtt{in} \downarrow ab_2.\mathtt{in} \downarrow cb_2.\mathtt{out} \uparrow ab.\mathtt{out} \uparrow cb)]
\end{array}
$$

Figure 12: An embedding of CCSP.

5 Behavioral Semantics

We define a compositional behavioral semantics of the conversation calculus by means of strong bisimulation. The main technical results of this section are the proofs that strong and weak bisimilarity are congruences for all the primitives of our calculus. This further ensures that our syntactically defined constructions induce properly defined behavioral operators at the semantic level.

Definition 5.1 *A (strong) bisimulation is a symmetric binary relation $\mathcal{R}$ on processes such that, for all processes P and Q, if $P\mathcal{R}Q$, we have:*

If $P \xrightarrow{\lambda} P'$ and $bn(\lambda) \cap fn(Q) = \emptyset$ then there is a process Q' such that $Q \xrightarrow{\lambda} Q'$ and $P'\mathcal{R}Q'$.

We denote by $\sim$ (strong bisimilarity) the largest strong bisimulation.

Strong bisimilarity is an equivalence relation. We also have:

Theorem 5.2 *Strong bisimilarity is a congruence for all operators.*

1. $P \mid \mathtt{stop} \sim P$.

[1] Late developments originate in a collaboration with Carla Ferreira, in particular the addition of the **try-catch** encoding.

2. $P \mid Q \sim Q \mid P$.
3. *If* $P \sim Q$ *then* $P \mid R \sim Q \mid R$.
4. *If* $P \sim Q$ *then* $!P \sim !Q$.
5. *If* $P \sim Q$ *then* $(\mathtt{new}\ a)P \sim (\mathtt{new}\ a)Q$.
6. *If* $P \sim Q$ *then* $\mathtt{throw}.P \sim \mathtt{throw}.Q$.
7. *If* $P\{x{\leftarrow}n\} \sim Q\{x{\leftarrow}n\}$ *for all* n *then* $\mathtt{here}(x).P \sim \mathtt{here}(x).Q$.
8. *If* $P \sim Q$ *then* $\mathtt{out}\ \alpha(v).P \sim \mathtt{out}\ \alpha(v).Q$.
9. *If* $P\{x{\leftarrow}n\} \sim Q\{x{\leftarrow}n\}$ *for all* n *then* $\mathtt{in}\ \alpha(x).P \sim \mathtt{in}\ \alpha(x).Q$.
10. *If* $P \sim Q$ *then* $\mathtt{try}\ P\ \mathtt{catch}\ R \sim \mathtt{try}\ Q\ \mathtt{catch}\ R$.
11. *If* $P \sim Q$ *then* $\mathtt{try}\ R\ \mathtt{catch}\ P \sim \mathtt{try}\ R\ \mathtt{catch}\ Q$.
12. *If* $P \sim Q$ *then* $n \blacktriangleright [P] \sim n \blacktriangleright [Q]$.
13. *If* $P \sim Q$ *then* $\mathtt{def}\ s \Rightarrow P \sim \mathtt{def}\ s \Rightarrow Q$.
14. *If* $P \sim Q$ *then* $\mathtt{instance}\ n\ \rho\ s \Rightarrow P \sim \mathtt{instance}\ n\ \rho\ s \Rightarrow Q$.

We consider weak bisimilarity defined as usual, denoted by $\approx$.

Theorem 5.3 *Weak bisimilarity is a congruence for all operators.*

Proof. Follows the lines of the proof of Theorem 5.2.

We also prove other interesting behavioral equations.

Proposition 5.4 *The following equations hold up to strong bisimilarity.*

1. *If* $n \neq m$ *then* $n \blacktriangleright [(\mathtt{new}\ m)P] \sim (\mathtt{new}\ m)n \blacktriangleright [P]$.
2. $n \blacktriangleright [P] \mid n \blacktriangleright [Q] \sim n \blacktriangleright [P \mid Q]$.
3. $m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [P]]] \sim n \blacktriangleright [o \blacktriangleright [P]]$.
4. $n \blacktriangleright [\mathtt{stop}] \sim \mathtt{stop}$.
5. $n \blacktriangleright [\mathtt{out} \uparrow l(\widetilde{v}).R] \sim \mathtt{out} \downarrow l(\widetilde{v}).n \blacktriangleright [R]$.
6. $n \blacktriangleright [\mathtt{in} \uparrow l(\widetilde{x}).R] \sim \mathtt{in} \downarrow l(\widetilde{x}).n \blacktriangleright [R]$ *(*$n \notin \widetilde{x}$*).*
7. $m \blacktriangleright [n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).P]] \sim n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [P]]]$.
8. $m \blacktriangleright [n \blacktriangleright [\mathtt{in} \downarrow l(\widetilde{x}).P]] \sim n \blacktriangleright [\mathtt{in} \downarrow l(\widetilde{x}).m \blacktriangleright [n \blacktriangleright [P]]]$ *(*$m,n \notin \widetilde{x}$*).*
9. $m \blacktriangleright [n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).P]] \sim n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [P]]]$.
10. $m \blacktriangleright [n \blacktriangleright [\mathtt{in} \leftarrow l(\widetilde{x}).P]] \sim n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{x}).m \blacktriangleright [n \blacktriangleright [P]]]$ *(*$m,n \notin \widetilde{x}$*).*
11. $m \blacktriangleright [n \blacktriangleright [\mathtt{def}\ s \Rightarrow P]] \sim n \blacktriangleright [\mathtt{def}\ s \Rightarrow P]$
12. $m \blacktriangleright [n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow P]] \sim n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow P]$

(Note: ▶ or ◀)

For instance, Proposition 5.4(3) captures the local character of message-based communication in our model. The behavioral identities stated in Proposition 5.4 allow us to prove an interesting normal form property, that contributes to illuminate the spatial structure of conversation context systems. A guarded process is a process of the form $\mathtt{out}\ \alpha(\widetilde{v}).P$ or $\mathtt{in}\ \alpha(\widetilde{x}).P$, $\mathtt{here}(x).P$, $\mathtt{instance}\ n\ \rho\ s \Leftarrow P$, or $\mathtt{def}\ s \Rightarrow P$. We use G to range over parallel compositions of guarded processes. We then have

Proposition 5.5 *Let P be a process in the finite exception-free fragment. Then there exist sets of guarded processes $\widetilde{G}, \widetilde{G'}, \widetilde{G''}$, sets of names $\widetilde{a}, \widetilde{b}, \widetilde{c}, \widetilde{d}$, and roles $\widetilde{\rho}, \widetilde{\rho'}, \widetilde{\rho''}$ such that*

$$P \sim (\mathtt{new}\ \widetilde{a})(\quad G_1 \mid \ldots \mid G_t \mid b_1\ \rho_1\ [G'_1] \mid \ldots \mid b_j\ \rho_j \left[G'_j\right] \\ \mid c_1\ \rho'_1\ [d_1\ \rho''_1\ [G''_1]] \mid \ldots \mid c_k\ \rho'_k \left[d_k\ \rho''_k \left[G''_k\right]\right])$$

and where the sequences (b_i, ρ_i) and $(c_i, \rho'_i, d_i, \rho''_i)$ are pairwise distinct.

Intuitively, Proposition 5.5 states that any process (of the finite exception-free fragment of the calculus) is behaviorally equivalent to a process where the maximum nesting of contexts is two. The restriction to finite (replication-free) exception-free processes is sensible, if one just wants to focus on the communication topology.

We interpret the normal form result as follows. A system is composed by several conversation contexts. The upward (↑) communication paths of a system may be seen as a graph, where arcs connect processes to their caller contexts. As each such arc is uniquely defined by its two terminal nodes, so is the communication structure of an arbitrary process defined (up to bisimilarity) by a system of (at most) depth two.

6 Related work

Various calculi have been recently proposed with the aim to capture aspects of service-oriented computation. At the root of each one, one finds different motivations and methodological approaches. Some intend to model artifacts of the web services technology, in order to develop applied verification techniques (e.g., COWS [18], SOCK [11]), others were introduced in order to demonstrate analysis techniques (e.g., [6, 7]), yet others have the goal of isolating primitives for formalizing and programming service-oriented applications (SCC [2], SSCC [17], CaSPiS [3]) just to refer a few.

The inspiration for the work presented here was motivated by previous developments around SCC [2], a process calculus designed to model service-oriented computing introduced within the Sensoria Project [14]. Our proposal inherits from [12] and SCC the presence of client-server session establishment primitives. However, we end up following a fresh approach, based on the notion of conversation context, and on a simple and flexible message-passing communication. Our development of the concept of conversation context was initially motivated by the concept of session (see [12]). We see conversation contexts as being more general than sessions, in the same sense that coroutining may be seen as a generalization of the stricter procedure (stack-oriented) call discipline. Moreover, the fact that in our model endpoint accesses may appear as arbitrary interacting processes to their enclosing contexts makes them quite different from the more familiar data streaming session endpoints.

Our up (↑) communication primitive was introduced with the aim of expressing the interaction between nested conversation contexts, in particular, between service instances endpoints and their callers, with loose-coupling in mind. Similar primitives have been already introduced

in ambient calculi, namely Seal [8], Boxed Ambients [4] and Box π [22]. Our computation model is very different from those models (which are targeted at modeling migration and mobility), as witnessed by Proposition 5.5. Hence, even if formally related to some primitives introduced in [4, 8], at least when their reaction rules are considered in isolation, our communication primitives have very different consequences at the semantic level (for example, two $\uparrow$ messages can synchronize, just as long as they originate in subcontexts of the same context).

Primitives to deal with exceptional behavior (for example, closing sessions) are present in several service calculi. Perhaps surprisingly, our exception mechanism, although clearly based on the classical construct for functional languages, does not seem to have been much explored in process calculi; we believe that it allows us to express many interesting exceptional behavior situations.

We have demonstrated that our approach is expressive enough to capture Orc's composition operators; we expect that similar results may be established for calculi with related constructs, such as streams and pipelines [17, 3], at least in the absence of types.

7 Concluding Remarks

We have presented a model for service-oriented computation, building on the identification of some general aspects of service-based systems. We have instantiated our model by proposing the conversation calculus, which incorporates abstractions of the several aspects involved by means of carefully chosen programming language primitives. We have focused our presentation on a detailed justification of the concepts involved, on examples that illustrate the expressiveness of our model, and on the semantic theory for our calculus, based on a standard strong bisimilarity. Our examples demonstrate how our calculus may express many service-oriented idioms in a rather natural way. The behavioral semantics allowed us to prove several interesting behavioral identities. Some of these identities suggested a normal form result that clarifies the spatial communication topology of conversation calculus systems.

Conversation contexts are natural subjects for typing disciplines, in terms of the message interchange patterns that may happen at their borders. We expect types specifying various properties of interfaces, service contracts, endpoint session protocols, security policies, resource usage, and service level agreements, to be in general assigned to context boundaries. One of the most interesting challenges to be addressed by type systems for the conversation calculus is then to discipline the delegation of conversation contexts according to quite strict usage disciplines, allowing for the static verification of systems where several (not just two) partners join and leave dynamically a conversation in a coordinated way.

Acknowledgments We thank our colleagues of the Sensoria Project for many discussions about programming language concepts and calculi for service based computing.

References

[1] A. Alves and et al. Web Services Business Process Execution Language Version 2.0. Technical report, OASIS, 2006.

[2] M. Boreale, R. Bruni, L. Caires, R. De Nicola, I. Lanese, M. Loreti, F. Martins, U. Montanari, A. Ravara, D. Sangiorgi, V. Vasconcelos, and G. Zavattaro. SCC: a Service Centered Calculus. In *Proceedings of WS-FM 2006, 3rd International Workshop on Web Services and Formal Methods*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[3] M. Boreale, R. Bruni, R. De Nicola, and M. Loreti. A Service Oriented Process Calculus with Sessioning and Pipelining. Technical report, 2007. Draft.

[4] M. Bugliesi, G. Castagna, and S. Crafa. Access Control for Mobile Agents: The Calculus of Boxed Ambients. *ACM Transactions on Programming Languages and Systems*, 26(1):57–124, 2004.

[5] M. J. Butler, C. A. R. Hoare, and C. Ferreira. A Trace Semantics for Long-Running Transactions. In A. E. Abdallah, C. B. Jones, and J. W. Sanders, editors, *25 Years Communicating Sequential Processes*, volume 3525 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2004.

[6] L. Caires. Spatial-Behavioral Types for Distributed Services and Resources. In U. Montanari and D. Sanella, editors, *Proceedings of the Second International Symposium on Trustworthy Global Computing*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[7] M. Carbone, K. Honda, and N. Yoshida. Structured Global Programming for Communication Behavior. In R. De Nicola, editor, *Proceedings of 16th European Symposium on Programming (ESOP'07)*, Lecture Notes in Computer Science. Springer, 2007.

[8] G. Castagna, J. Vitek, and F. Z. Nardelli. The Seal Calculus. *Information and Computation*, 201(1):1–54, 2005.

[9] J. L. Fiadeiro, A. Lopes, and L. Bocchi. A Formal Approach to Service Component Architecture. In M. Bravetti, M. N., and G. Zavattaro, editors, *Web Services and Formal Methods, Third International Workshop,WS-FM 2006*, volume 4184 of *Lecture Notes in Computer Science*, pages 193–213. Springer-Verlag, 2006.

[10] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

[11] C. Guidi, R. Lucchi, R. Gorrieri, N. Busi, and G. Zavattaro. SOCK: A Calculus for Service Oriented Computing. In M. Bravetti, M. N., and G. Zavattaro, editors, *Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC 2006)*, volume 4294 of *Lecture Notes in Computer Science*, pages 327–338. Springer-Verlag, 2006.

[12] K. Honda, V. T. Vasconcelos, and M. Kubo. Language Primitives and Type Discipline for Structured Communication-Based Programming. In C. Hankin, editor, *ESOP'98, 7th European Symposium on Programming,ETAPS'98*, volume 1381 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 1998.

[13] Amazon.com Inc. Amazon Simple Storage Service Developer Guide, 2007. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/.

[14] IP Sensoria Project. website: http://www.sensoria-ist.eu/.

[15] M. B. Juric. A Hands-on Introduction to BPEL, 2006. Oracle (white paper).

[16] D. Kitchin, W. R. Cook, and J. Misra. A Language for Task Orchestration and Its Semantic Properties. In C. Baier and H. Hermanns, editors, *CONCUR 2006 - Concurrency Theory, 17th International Conference*, volume 4137 of *Lecture Notes in Computer Science*, pages 477–491. Springer-Verlag, 2006.

[17] I. Lanese, V. T. Vasconcelos, F. Martins, and A. Ravara. Disciplining Orchestration and Conversation in Service-Oriented Computing. In *5th International Conference on Software Engineering and Formal Methods*, pages 305–314. IEEE Computer Society Press, 2007.

[18] A. Lapadula, R. Pugliese, and F. Tiezzi. A Calculus for Orchestration of Web Services. In R. De Nicola, editor, *Proceedings of 16th European Symposium on Programming (ESOP'07)*, Lecture Notes in Computer Science. Springer, 2007.

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

[20] J. Misra and W. R. Cook. Computation Orchestration: A Basis for Wide-Area Computing. *Journal of Software and Systems Modeling*, 2006.

[21] D. Sangiorgi and D. Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[22] P. Sewell and J. Vitek. Secure Composition of Untrusted Code: Box π, Wrappers, and Causality. *Journal of Computer Security*, 11(2):135–188, 2003.

A Proofs

Proof of auxiliar results to Theorem 5.2

Lemma A.1 *Let P, Q be processes such that $P \sim Q$. Then* $\texttt{throw}.P \sim \texttt{throw}.Q$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(\texttt{throw}.P, \texttt{throw}.Q) \mid P \sim Q\} \tag{A.1.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \sim$ and hence $(P', Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.1.1), of the form $\texttt{throw}.\bar{P}$ and $\texttt{throw}.\bar{Q}$, respectively, for some $\bar{P}, \bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.1.2}$$

We have that the only possible transition of $\texttt{throw}.\bar{P}$ is

$$\texttt{throw}.\bar{P} \xrightarrow{\texttt{throw}} \bar{P}$$

We also have that

$$\texttt{throw}.\bar{Q} \xrightarrow{\texttt{throw}} \bar{Q}$$

From A.1.2 we conclude that $(\bar{P}, \bar{Q}) \in \mathcal{R} \cup \sim$ which completes the proof.

Lemma A.2 *Let P, Q be processes such that $P \sim Q$. Then for any pair of location name α and set of names $\widetilde{v}$ it is the case that* $\texttt{out}\ \alpha(\widetilde{v}).P \sim \texttt{out}\ \alpha(\widetilde{v}).Q$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(\texttt{out}\ \alpha(\widetilde{v}).P, \texttt{out}\ \alpha(\widetilde{v}).Q) \mid P \sim Q \wedge \forall \alpha, \widetilde{v}\} \tag{A.2.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.2.1), of the form $\mathtt{out}\ \alpha(\widetilde{v}).\bar{P}$ and $\mathtt{out}\ \alpha(\widetilde{v}).\bar{Q}$, respectively, for some $\alpha, \widetilde{v}, \bar{P}, \bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.2.2}$$

We have that the only possible transition of $\mathtt{out}\ \alpha(\widetilde{v}).\bar{P}$ is

$$\mathtt{out}\ \alpha(\widetilde{v}).\bar{P} \xrightarrow{\overline{\alpha(\widetilde{v})}} \bar{P}$$

We also have that

$$\mathtt{out}\ \alpha(\widetilde{v}).\bar{Q} \xrightarrow{\overline{\alpha(\widetilde{v})}} \bar{Q}$$

From A.2.2 we conclude that $(\bar{P},\bar{Q}) \in \mathcal{R} \cup \sim$ which completes the proof.

Lemma A.3 *Let P,Q be processes such that for all set of variables $\widetilde{x}$ and set of names $\widetilde{v}$ it is the case that $P\{\widetilde{x} \leftarrow \widetilde{v}\} \sim Q\{\widetilde{x} \leftarrow \widetilde{v}\}$. Then for any pair of location name α it is the case that $\mathtt{in}\ \alpha(\widetilde{x}).P \sim \mathtt{in}\ \alpha(\widetilde{x}).Q$.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(\mathtt{in}\ \alpha(\widetilde{x}).P, \mathtt{in}\ \alpha(\widetilde{x}).Q) \mid \forall \widetilde{x}, \widetilde{v}\ .\ P\{\widetilde{x} \leftarrow \widetilde{v}\} \sim Q\{\widetilde{x} \leftarrow \widetilde{v}\}\} \tag{A.3.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.3.1), of the form $\mathtt{in}\ \alpha(\widetilde{x}).\bar{P}$ and $\mathtt{in}\ \alpha(\widetilde{x}).\bar{Q}$, respectively, for some $\alpha, \widetilde{x}, \bar{P}, \bar{Q}$ such that for all $\widetilde{v}$ it is the case that

$$\bar{P}\{\widetilde{x} \leftarrow \widetilde{v}\} \sim \bar{Q}\{\widetilde{x} \leftarrow \widetilde{v}\} \tag{A.3.2}$$

We have that $\mathtt{in}\ \alpha(\widetilde{x}).\bar{P}$ has for any $\widetilde{v}$ a transition

$$\mathtt{in}\ \alpha(\widetilde{x}).\bar{P} \xrightarrow{\alpha(\widetilde{v})} \bar{P}\{\widetilde{x} \leftarrow \widetilde{v}\}$$

We also have that

$$\mathtt{in}\ \alpha(\widetilde{x}).\bar{Q} \xrightarrow{\alpha(\widetilde{v})} \bar{Q}\{\widetilde{x} \leftarrow \widetilde{v}\}$$

From A.3.2 we conclude that for any $\widetilde{v}$ it is the case that $(\bar{P}\{\widetilde{x} \leftarrow \widetilde{v}\}, \bar{Q}\{\widetilde{x} \leftarrow \widetilde{v}\}) \in \mathcal{R} \cup \sim$ which completes the proof.

Lemma A.4 *Let P,Q be processes such that for any name n it is the case that $P\{x\leftarrow n\} \sim Q\{x\leftarrow n\}$. Then for any ρ it is the case that* $\texttt{here}\ \rho(x).P \sim \texttt{here}\ \rho(x).Q$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(\texttt{here}\ \rho(x).P, \texttt{here}\ \rho(x).Q) \mid \forall n\ .\ P\{x\leftarrow n\} \sim Q\{x\leftarrow n\}\} \tag{A.4.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.4.1), of the form $\texttt{here}\ \rho(x).\bar{P}$ and $\texttt{here}\ \rho(x).\bar{Q}$, respectively, for some $\rho, x, \bar{P}, \bar{Q}$ such that for any name n it is the case that

$$\bar{P}\{x\leftarrow n\} \sim \bar{Q}\{x\leftarrow n\} \tag{A.4.2}$$

We have that $\texttt{here}\ \rho(x).\bar{P}$ has for any name n a transition

$$\texttt{here}\ \rho(x).\bar{P} \xrightarrow{n\rho\texttt{here}} \bar{P}\{x\leftarrow n\}$$

We also have that

$$\texttt{here}\ \rho(x).\bar{Q} \xrightarrow{n\rho\texttt{here}} \bar{Q}\{x\leftarrow n\}$$

From A.4.2 we conclude that for any name n it is the case that $(\bar{P}\{x\leftarrow n\}, \bar{Q}\{x\leftarrow n\}) \in \mathcal{R} \cup \sim$ which completes the proof.

Lemma A.5 *Let P,Q be processes such that $P \sim Q$. Then for any name n it is the case that $n \blacktriangleright [P] \sim n \blacktriangleright [Q]$.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(n \blacktriangleright [P], n \blacktriangleright [Q]) \mid P \sim Q\} \tag{A.5.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have, by definition (A.5.1), that P and Q are of the form $n \blacktriangleright [\bar{P}]$ and $n \blacktriangleright \left[\bar{Q}\right]$, respectively, for some $n, \bar{P}, \bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.5.2}$$

We must consider eight different derivations for the transitions of $n \blacktriangleright [P]$, so we have that λ is either derived from τ, or from `throw`, or from $(\widetilde{\nu o})m\rho\lambda'$ such that $\lambda' \neq \texttt{here}$, or from $n \blacktriangleright \texttt{here}$, or from $(\widetilde{\nu o})\lambda'^{\leftarrow}$, or from $(\widetilde{\nu o})\lambda'^{\downarrow}$, or from $(\widetilde{\nu o})\lambda'^{\uparrow}$ or finally from $(\nu c)\texttt{def}\ s$.

(*Case* τ)

We have that

$$n \blacktriangleright [\bar{P}] \xrightarrow{\tau} \bar{P}' \tag{A.5.3}$$

where (A.5.3) is derived from

$$\bar{P} \xrightarrow{\tau} \bar{P}' \tag{A.5.4}$$

From (A.5.4) and (A.5.2) we have that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{\tau} \bar{Q}' \tag{A.5.5}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.5.6}$$

From (A.5.5) we can derive

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} n \blacktriangleright \left[\bar{Q}'\right]$$

From (A.5.6) we conclude

$$\left(n \blacktriangleright \left[\bar{P}'\right], n \blacktriangleright \left[\bar{Q}'\right]\right) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* `throw`)

We have that

$$n \blacktriangleright [\bar{P}] \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.5.7}$$

where (A.5.7) is derived from

$$\bar{P} \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.5.8}$$

From (A.5.8) and (A.5.2) we have that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{\texttt{throw}} \bar{Q}' \tag{A.5.9}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.5.10}$$

From (A.5.9) we can derive

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\texttt{throw}} \bar{Q}'$$

From (A.5.10) we conclude

$$(\bar{P}', \bar{Q}') \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $((\widetilde{\nu o})m\rho\lambda')$

We have that

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})m\rho\lambda'} n \blacktriangleright \left[\bar{P}'\right] \tag{A.5.11}$$

where (A.5.11) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})m\rho\lambda'} \bar{P}' \tag{A.5.12}$$

and $\lambda' \neq \texttt{here}$. From (A.5.12) and (A.5.2) we have that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{(\widetilde{\nu o})m\rho\lambda'} \bar{Q}' \tag{A.5.13}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.5.14}$$

From (A.5.13) recalling that $\lambda' \neq \texttt{here}$ we can derive

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})m\rho\lambda'} n \blacktriangleright \left[\bar{Q}'\right]$$

From (A.5.14) and by definition of $\mathcal{R}$ (A.5.1) we conclude

$$\left(n \blacktriangleright \left[\bar{P}'\right], n \blacktriangleright \left[\bar{Q}'\right]\right) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $n \blacktriangleright \texttt{here}$)

We have that

$$n \blacktriangleright [\bar{P}] \xrightarrow{\tau} n \blacktriangleright \left[\bar{P}'\right] \tag{A.5.15}$$

where (A.5.15) is derived from

$$\bar{P} \xrightarrow{n \blacktriangleright \texttt{here}} \bar{P}' \tag{A.5.16}$$

From (A.5.16) and (A.5.2) we have that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{n \blacktriangleright \texttt{here}} \bar{Q}' \tag{A.5.17}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.5.18}$$

From (A.5.17) we can derive

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} n \blacktriangleright \left[\bar{Q}'\right]$$

From (A.5.18) and by definition of $\mathcal{R}$ (A.5.1) we conclude

$$\left(n \blacktriangleright \left[\bar{P}'\right], n \blacktriangleright \left[\bar{Q}'\right]\right) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $\lambda = (\widetilde{\nu o})\lambda'^{\leftarrow}$)

We have that

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n \blacktriangleright \cdot \lambda'^{\leftarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.5.19}$$

where (A.5.19) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda'^{\leftarrow}} \bar{P}' \tag{A.5.20}$$

From (A.5.20) and (A.5.2) we have that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{(\widetilde{\nu o})\lambda'^{\leftarrow}} \bar{Q}' \tag{A.5.21}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.5.22}$$

From (A.5.21) we can derive

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\leftarrow}} n \blacktriangleright \left[\bar{Q}'\right]$$

From (A.5.22) and by definition of $\mathcal{R}$ (A.5.1) we conclude

$$\left(n \blacktriangleright \left[\bar{P}'\right], n \blacktriangleright \left[\bar{Q}'\right]\right) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $\lambda = (\widetilde{\nu o})\lambda'^{\downarrow}$)

We have that

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.5.23}$$

where (A.5.23) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} \bar{P}' \tag{A.5.24}$$

From (A.5.24) and (A.5.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} \bar{Q}' \tag{A.5.25}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.5.26}$$

From (A.5.25) we can derive

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{Q}'\right]$$

From (A.5.26) and by definition of $\mathcal{R}$ (A.5.1) we conclude

$$\left(n \blacktriangleright \left[\bar{P}'\right], n \blacktriangleright \left[\bar{Q}'\right]\right) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $\lambda = (\widetilde{\nu o})\lambda'^{\uparrow}$)

We have that

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.5.27}$$

where (A.5.27) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda'^{\uparrow}} \bar{P}' \tag{A.5.28}$$

From (A.5.28) and (A.5.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{(\widetilde{\nu o})\lambda'^{\uparrow}} \bar{Q}' \tag{A.5.29}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.5.30}$$

From (A.5.29) we can derive

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{Q}'\right]$$

From (A.5.30) and by definition of $\mathcal{R}$ (A.5.1) we conclude

$$\left(n \blacktriangleright \left[\bar{P}'\right], n \blacktriangleright \left[\bar{Q}'\right]\right) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $\lambda = (\nu c)\texttt{def}\ s$)

We have that

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\nu c)n \blacktriangleright \texttt{def}\ s} n \blacktriangleright [\bar{P}'] \tag{A.5.31}$$

where (A.5.31) is derived from

$$\bar{P} \xrightarrow{(\nu c)\texttt{def}\ s} \bar{P}' \tag{A.5.32}$$

From (A.5.32) and (A.5.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{(\nu c)\texttt{def}\ s} \bar{Q}' \tag{A.5.33}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.5.34}$$

From (A.5.33) we can derive

$$n \blacktriangleright [\bar{Q}] \xrightarrow{(\nu c)n \blacktriangleright \texttt{def}\ s} n \blacktriangleright [\bar{Q}']$$

From (A.5.34) and by definition of $\mathcal{R}$ (A.5.1) we conclude

$$(n \blacktriangleright [\bar{P}'], n \blacktriangleright [\bar{Q}']) \in \mathcal{R} \cup \sim$$

which completes the proof for this last case.

Lemma A.6 *Let P, Q be processes such that $P \sim Q$. Then for any name n it is the case that $n \blacktriangleleft [P] \sim n \blacktriangleleft [Q]$.*

Proof. Analogous to that of Lemma A.5.

Lemma A.7 *Let P, Q be processes such that $P \sim Q$. Then for any name s it is the case that $\texttt{def}\ s \Rightarrow P \sim \texttt{def}\ s \Rightarrow Q$.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(\texttt{def}\ s \Rightarrow P, \texttt{def}\ s \Rightarrow Q) \mid P \sim Q\} \tag{A.7.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P, Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P, Q) \in \mathcal{R}$ or $(P, Q) \in \sim$.

If $(P, Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \sim$ and hence $(P', Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.7.1), of the form $\mathtt{def}\ s \Rightarrow \bar{P}$ and $\mathtt{def}\ s \Rightarrow \bar{Q}$, respectively, for some name s and processes $\bar{P}, \bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.7.2}$$

We have that $\mathtt{def}\ s \Rightarrow \bar{P}$ has for any c a possible transition

$$\mathtt{def}\ s \Rightarrow \bar{P} \xrightarrow{(\nu c)\mathtt{def}\ s} c \blacktriangleright [\bar{P}]$$

which we only consider when the bound name generated does not occur in process Q, hence $c \notin fn(Q)$, accordingly to the definition of bisimulation. Given that $c \notin fn(Q)$ we also have that

$$\mathtt{def}\ s \Rightarrow \bar{Q} \xrightarrow{(\nu c)\mathtt{def}\ s} c \blacktriangleright \left[\bar{Q}\right]$$

From (A.7.2) and considering Lemma A.5 we have that

$$(c \blacktriangleright [\bar{P}], c \blacktriangleright \left[\bar{Q}\right]) \in \sim$$

hence

$$(c \blacktriangleright [\bar{P}], c \blacktriangleright \left[\bar{Q}\right]) \in \mathcal{R} \cup \sim$$

which completes the proof.

Lemma A.8 *Let P,Q be processes such that $P \sim Q$. Then for any names n,s it is the case that* $\mathtt{instance}\ n \blacktriangleright s \Leftarrow P \sim \mathtt{instance}\ n \blacktriangleright s \Leftarrow Q$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(\mathtt{instance}\ n \blacktriangleright s \Leftarrow P, \mathtt{instance}\ n \blacktriangleright s \Leftarrow Q) \mid P \sim Q\} \tag{A.8.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.8.1), of the form $\mathtt{instance}\ n \blacktriangleright s \Leftarrow \bar{P}$ and $\mathtt{instance}\ n \blacktriangleright s \Leftarrow \bar{Q}$, respectively, for some names n,s and processes $\bar{P}, \bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.8.2}$$

We have that $\mathtt{instance}\ n \blacktriangleright s \Leftarrow \bar{P}$ has for any c a possible transition

$$\mathtt{instance}\ n \blacktriangleright s \Leftarrow \bar{P} \xrightarrow{\overline{(\nu c)n \blacktriangleright \mathtt{def}\ s}} c \blacktriangleleft [\bar{P}]$$

which we only consider when the bound name generated does not occur in process Q, hence $c \notin fn(Q)$, accordingly to the definition of bisimulation. Given that $c \notin fn(Q)$ we also have that

$$\texttt{instance}\ n \blacktriangleright s \Leftarrow \bar{Q} \xrightarrow{\overline{(\nu c)n\blacktriangleright \texttt{def}\ s}} c \blacktriangleleft \left[\bar{Q}\right]$$

From (A.8.2) and considering Lemma A.6 we have that

$$(c \blacktriangleleft [\bar{P}], c \blacktriangleleft \left[\bar{Q}\right]) \in \sim$$

hence

$$(c \blacktriangleleft [\bar{P}], c \blacktriangleleft \left[\bar{Q}\right]) \in \mathcal{R} \cup \sim$$

which completes the proof.

Lemma A.9 *Let P, Q be processes such that $P \sim Q$. Then for any names n, s it is the case that* $\texttt{instance}\ n \blacktriangleleft s \Leftarrow P \sim \texttt{instance}\ n \blacktriangleleft s \Leftarrow Q$.

Proof. Analogous to that of Lemma A.8.

Lemma A.10 *We have that $P \mid \texttt{stop} \sim P$, for any process P.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(P \mid \texttt{stop}, P)\} \cup \{(P, P)\} \tag{A.10.1}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P, Q) \in \mathcal{R}$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R}$.

We must consider two different cases: either $(P, Q) \in \{(P \mid \texttt{stop}, P)\}$ or $(P, Q) \in \{(P, P)\}$.

If $(P, Q) \in \{(P, P)\}$ we we directly have that for any transition $P \xrightarrow{\lambda} P'$ it is the case that $(P', P') \in \mathcal{R}$.

If $(P, Q) \in \{(P \mid \texttt{stop}, P)\}$ we have that P is of the form $Q \mid \texttt{stop}$.

We consider two possible transitions of $Q \mid \texttt{stop}$: either $\lambda \neq \texttt{throw}$ or $\lambda = \texttt{throw}$.

If $\lambda \neq \texttt{throw}$ we have that there exists Q' such that

$$Q \mid \texttt{stop} \xrightarrow{\lambda} Q' \mid \texttt{stop}$$

derived from

$$Q \xrightarrow{\lambda} Q'$$

which along with $(Q' \mid \texttt{stop}, Q') \in \mathcal{R}$ completes the proof for this case.

If $\lambda = \texttt{throw}$ we have that there exists Q' such that

$$Q \mid \texttt{stop} \xrightarrow{\texttt{throw}} Q'$$

derived from

$$Q \xrightarrow{\texttt{throw}} Q'$$

which along with $(Q', Q') \in \mathcal{R}$ completes the proof for this case.

We now consider two possible transitions for Q: either $\lambda \neq \texttt{throw}$ or $\lambda = \texttt{throw}$.

If $\lambda \neq \texttt{throw}$ we have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

from which we can derive

$$Q \mid \mathbf{stop} \xrightarrow{\lambda} Q' \mid \mathbf{stop}$$

which along with $(Q' \mid \mathbf{stop}, Q') \in \mathcal{R}$ completes the proof for this case.

If $\lambda = \texttt{throw}$ we have that there exists Q' such that

$$Q \xrightarrow{\texttt{throw}} Q'$$

from which we can derive

$$Q \mid \mathbf{stop} \xrightarrow{\texttt{throw}} Q'$$

which along with $(Q', Q') \in \mathcal{R}$ completes the proof for this last case.

Lemma A.11 *Let P, Q be processes such that $P \sim Q$. Then for any name n it is the case that $(\mathbf{new}\, n)P \sim (\mathbf{new}\, n)Q$.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{((\mathbf{new}\, n)P, (\mathbf{new}\, n)Q) \mid P \sim Q\} \tag{A.11.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P, Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P, Q) \in \mathcal{R}$ or $(P, Q) \in \sim$.

If $(P, Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \sim$ and hence $(P', Q') \in \mathcal{R} \cup \sim$.

If $(P, Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.11.1), of the form $(\mathbf{new}\, n)\bar{P}$ and $(\mathbf{new}\, n)\bar{Q}$, respectively, for some name n and processes $\bar{P}, \bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.11.2}$$

We have that $(\mathbf{new}\, n)\bar{P}$ has two possible transitions λ, distinguished by the occurrence of n as an extruded name of λ or not.

If λ extrudes n we have that

$$(\mathbf{new}\, n)\bar{P} \xrightarrow{(n)\lambda} \bar{P}' \tag{A.11.3}$$

where (A.11.3) is derived from

$$\bar{P} \xrightarrow{\lambda} \bar{P}' \tag{A.11.4}$$

and $n \in out(\lambda)$. From (A.11.4) and (A.11.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{\lambda} \bar{Q}' \tag{A.11.5}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.11.6}$$

From (A.11.5) and the fact that $n \in out(\lambda)$ we conclude that

$$(\mathbf{new}\, n)\bar{Q} \xrightarrow{(n)\lambda} \bar{Q}'$$

From (A.11.6) we directly have that

$$(\bar{P}', \bar{Q}') \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

If λ does not extrude n we have that

$$(\mathbf{new}\, n)\bar{P} \xrightarrow{\lambda} (\mathbf{new}\, n)\bar{P}' \tag{A.11.7}$$

where (A.11.7) is derived from

$$\bar{P} \xrightarrow{\lambda} \bar{P}' \tag{A.11.8}$$

and $n \notin out(\lambda)$. From (A.11.8) and (A.11.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{\lambda} \bar{Q}' \tag{A.11.9}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.11.10}$$

From (A.11.9) and the fact that $n \notin out(\lambda)$ we conclude that

$$(\mathbf{new}\, n)\bar{Q} \xrightarrow{\lambda} (\mathbf{new}\, n)\bar{Q}'$$

From (A.11.10) we conclude that

$$((\mathbf{new}\, n)\bar{P}', (\mathbf{new}\, n)\bar{Q}') \in \mathcal{R} \cup \sim$$

which completes the proof for this last case.

Lemma A.12 *We have $P \mid Q \sim Q \mid P$, for any processes P, Q.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{((\mathbf{new}\, \widetilde{n})(P \mid Q), (\mathbf{new}\, \widetilde{n})(Q \mid P)) \mid \forall \widetilde{n}\} \cup \{(P,P)\} \tag{A.12.1}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R}$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R}$.

We must consider two different cases: either $(P,Q) \in \{((\mathbf{new}\,\widetilde{n})(P \mid Q), (\mathbf{new}\,\widetilde{n})(Q \mid P))\}$ or $(P,Q) \in \{(P,P)\}$.

If $(P,Q) \in \{(P,P)\}$ we we directly have that for any transition $P \xrightarrow{\lambda} P'$ it is the case that $(P',P') \in \mathcal{R}$.

If $(P,Q) \in \{((\mathbf{new}\,\widetilde{n})(P \mid Q), (\mathbf{new}\,\widetilde{n})(Q \mid P))\}$ we have that P and Q are of the form $(\mathbf{new}\,\widetilde{n})(\bar{P} \mid \bar{Q})$ and $(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid \bar{P})$, respectively, for some set of names $\widetilde{n}$ and processes $\bar{P}$ and $\bar{Q}$.

We must consider six different transitions for $(\mathbf{new}\,\widetilde{n})(\bar{P} \mid \bar{Q})$: either $\bar{P}$ performs a transition (either a `throw` or some other λ) or $\bar{Q}$ performs a transition (again either a `throw` or some other λ) or finally $\bar{P}$ and $\bar{Q}$ synchronize either by $n\texttt{phere}$ or by τ.

(*Transition* `throw` *performed by* $\bar{P}$)

If $\bar{P}$ performs a transition `throw` we have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid \bar{Q}) \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{n})\bar{P}' \tag{A.12.2}$$

where (A.12.2) is derived from

$$\bar{P} \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.12.3}$$

From (A.12.3) we can derive that

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid \bar{P}) \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{n})\bar{P}'$$

which along with

$$((\mathbf{new}\,\widetilde{n})\bar{P}', (\mathbf{new}\,\widetilde{n})\bar{P}') \in \mathcal{R}$$

completes the proof for this case.

(*Transition* λ *performed by* $\bar{P}$)

If $\bar{P}$ performs a transition λ, different from `throw`, we have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid \bar{Q}) \xrightarrow{\lambda} (\mathbf{new}\,\widetilde{n}')(\bar{P}' \mid \bar{Q}) \tag{A.12.4}$$

where (A.12.4) is derived from

$$\bar{P} \xrightarrow{\lambda'} \bar{P}' \tag{A.12.5}$$

having $\lambda = (\widetilde{\nu n''})\lambda'$, i.e., label λ is obtained by placing a set of bound names in front of λ', a set that corresponds exactly to the names that λ' extrudes that are restricted names contained in $\widetilde{n}$, hence $\widetilde{n}'' = out(\lambda') \cap \widetilde{n}$. Also we have that the resulting restricted name set $\widetilde{n}'$ corresponds to the initial one $\widetilde{n}$ minus the names extruded, hence $(\widetilde{\nu n'} = \widetilde{n} \,/\, \widetilde{n}'')$. We also have that $(bn(\lambda') \cap fn(\bar{Q}) = \emptyset)$. Attending to these conditions on the restricted names from (A.12.5) we can derive that

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid \bar{P}) \xrightarrow{\lambda} (\mathbf{new}\,\widetilde{n}')(\bar{Q} \mid \bar{P}')$$

which along with

$$((\mathbf{new}\,\widetilde{n}')(\bar{P}' \mid \bar{Q}), (\mathbf{new}\,\widetilde{n}')(\bar{Q} \mid \bar{P}')) \in \mathcal{R}$$

completes the proof for this case.

(*Transition* `throw` *performed by* $\bar{Q}$)

If $\bar{Q}$ performs a transition `throw` we have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid \bar{Q}) \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{n})\bar{Q}' \tag{A.12.6}$$

where (A.12.6) is derived from

$$\bar{Q} \xrightarrow{\texttt{throw}} \bar{Q}' \tag{A.12.7}$$

From (A.12.7) we can derive that

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid \bar{P}) \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{n})\bar{Q}'$$

which along with

$$((\mathbf{new}\,\widetilde{n})\bar{Q}', (\mathbf{new}\,\widetilde{n})\bar{Q}') \in \mathcal{R}$$

completes the proof for this case.

(*Transition* λ *performed by* $\bar{Q}$)

If $\bar{Q}$ performs a transition λ, different from `throw`, we have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid \bar{Q}) \xrightarrow{\lambda} (\mathbf{new}\,\widetilde{n}')(\bar{P} \mid \bar{Q}') \tag{A.12.8}$$

where (A.12.8) is derived from

$$\bar{Q} \xrightarrow{\lambda'} \bar{Q}' \tag{A.12.9}$$

having $\lambda = (\widetilde{\nu n}'')\lambda'$, i.e., label λ is obtained by placing a set of bound names in front of λ', a set that corresponds exactly to the names that λ' extrudes that are restricted names contained in $\widetilde{n}$, hence $\widetilde{n}'' = out(\lambda') \cap \widetilde{n}$. Also we have that the resulting restricted name set $\widetilde{n}'$ corresponds to the initial one $\widetilde{n}$ minus the names extruded, hence $(\widetilde{\nu n}' = \widetilde{n} \,/\, \widetilde{n}'')$. We also have that $(bn(\lambda') \cap fn(\bar{Q}) = \emptyset)$. Attending to these conditions on the restricted names from (A.12.9) we can derive that

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid \bar{P}) \xrightarrow{\lambda} (\mathbf{new}\,\widetilde{n}')(\bar{Q}' \mid \bar{P})$$

which along with

$$((\mathbf{new}\,\widetilde{n}')(\bar{P} \mid \bar{Q}'), (\mathbf{new}\,\widetilde{n}')(\bar{Q}' \mid \bar{P})) \in \mathcal{R}$$

completes the proof for this case.

(*Transition* $n\texttt{phere}$ *due to the synchronization of* $\bar{P}$ *and* $\bar{Q}$)

We have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid \bar{Q}) \xrightarrow{n\texttt{phere}} (\mathbf{new}\,\widetilde{n}, c)(\bar{P}' \mid \bar{Q}') \tag{A.12.10}$$

where (A.12.10) is derived from

$$\bar{P} \xrightarrow{(\nu c)\texttt{def}\ s} \bar{P}' \tag{A.12.11}$$

and

$$\bar{Q} \xrightarrow{(\nu c)\overline{n\texttt{phere}\ s}} \bar{Q}' \tag{A.12.12}$$

From (A.12.11) and (A.12.12) we conclude (using the symmetric rule)

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid \bar{P}) \xrightarrow{n\texttt{phere}} (\mathbf{new}\,\widetilde{n}, c)(\bar{Q}' \mid \bar{P}')$$

which along with

$$((\mathbf{new}\,\widetilde{n}, c)(\bar{Q}' \mid \bar{P}'), (\mathbf{new}\,\widetilde{n}, c)(\bar{Q}' \mid \bar{P}')) \in \mathcal{R}$$

completes the proof for this case.

(*Transition* τ *due to the synchronization of* $\bar{P}$ *and* $\bar{Q}$)

We have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid \bar{Q}) \xrightarrow{\tau} (\mathbf{new}\,\widetilde{n}, \widetilde{n}')(\bar{P}' \mid \bar{Q}') \tag{A.12.13}$$

where (A.12.13) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu n}')\lambda'} \bar{P}' \tag{A.12.14}$$

and

$$\bar{Q} \xrightarrow{(\widetilde{\nu n}')\overline{\lambda'}} \bar{Q}' \tag{A.12.15}$$

From (A.12.14) we can derive

$$\bar{P} \xrightarrow{(\widetilde{\forall n'})\overline{\overline{\lambda'}}} \bar{P}' \tag{A.12.16}$$

From (A.12.15) and (A.12.16) we conclude

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid \bar{P}) \xrightarrow{\tau} (\mathbf{new}\,\widetilde{n},\widetilde{n'})(\bar{Q}' \mid \bar{P}')$$

which along with

$$((\mathbf{new}\,\widetilde{n},\widetilde{n'})(\bar{Q}' \mid \bar{P}'),(\mathbf{new}\,\widetilde{n},\widetilde{n'})(\bar{Q}' \mid \bar{P}')) \in \mathcal{R}$$

completes the proof for this last case.

Lemma A.13 *Let P,Q be processes such that $P \sim Q$. Then for any process R it is the case that $P \mid R \sim Q \mid R$.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{((\mathbf{new}\,\widetilde{n})(P \mid R),(\mathbf{new}\,\widetilde{n})(Q \mid R)) \mid P \sim Q \wedge \forall \widetilde{n},R\} \tag{A.13.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P',λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.13.1), of the form $(\mathbf{new}\,\widetilde{n})(\bar{P} \mid R)$ and $(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R)$, respectively, for some set of names $\widetilde{n}$ and processes $R,\bar{P},\bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.13.2}$$

We must consider six different transitions for $(\mathbf{new}\,\widetilde{n})(\bar{P} \mid R)$: either $\bar{P}$ performs a transition (either a `throw` or some other λ) or R performs a transition (again either a `throw` or some other λ) or finally $\bar{P}$ and R synchronize, either by $n\rho\texttt{here}$ or by τ.

(*Transition* `throw` *performed by* $\bar{P}$)

If $\bar{P}$ performs a transition `throw` we have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid R) \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{n})\bar{P}' \tag{A.13.3}$$

where (A.13.3) is derived from

$$\bar{P} \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.13.4}$$

From (A.13.4) and (A.13.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{\texttt{throw}} \bar{Q}' \tag{A.13.5}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.13.6}$$

From (A.13.5) we can derive

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R) \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{n})\bar{Q}'$$

From (A.13.6) and considering Lemma A.11 we conclude

$$((\mathbf{new}\,\widetilde{n})\bar{P}', (\mathbf{new}\,\widetilde{n})\bar{Q}') \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Transition* λ *performed by* $\bar{P}$)

If $\bar{P}$ performs a transition λ, different from `throw`, we have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid R) \xrightarrow{\lambda} (\mathbf{new}\,\widetilde{n}')(\bar{P}' \mid R) \tag{A.13.7}$$

where (A.13.7) is derived from

$$\bar{P} \xrightarrow{\lambda'} \bar{P}' \tag{A.13.8}$$

having $\lambda = (\widetilde{\nu n''})\lambda'$, i.e., label λ is obtained by placing a set of bound names in front of λ', a set that corresponds exactly to the names that λ' extrudes that are restricted names contained in $\widetilde{n}$, hence $\widetilde{n}'' = out(\lambda') \cap \widetilde{n}$. Also we have that the resulting restricted name set $\widetilde{n}'$ corresponds to the initial one $\widetilde{n}$ minus the names extruded, hence $(\widetilde{\nu n}' = \widetilde{n} \,/\, \widetilde{n}'')$. We also have that $(bn(\lambda') \cap fn(R) = \emptyset)$. From (A.13.8) and (A.13.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{\lambda'} \bar{Q}' \tag{A.13.9}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.13.10}$$

Attending to conditions on the restricted names above from (A.13.9) we can derive

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R) \xrightarrow{\lambda} (\mathbf{new}\,\widetilde{n}')(\bar{Q}' \mid R)$$

From (A.13.10) and by definition of $\mathcal{R}$ (A.13.1) we conclude

$$((\mathbf{new}\,\widetilde{n}')(\bar{P}' \mid R), (\mathbf{new}\,\widetilde{n}')(\bar{Q}' \mid R)) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Transition* `throw` *performed by* R)

If R performs a transition `throw` we have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid R) \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{n})R' \tag{A.13.11}$$

where (A.13.11) is derived from

$$R \xrightarrow{\texttt{throw}} R' \tag{A.13.12}$$

From (A.13.12) we can derive

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R) \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{n})R'$$

We directly have that

$$((\mathbf{new}\,\widetilde{n})R', (\mathbf{new}\,\widetilde{n})R') \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Transition* λ *performed by* R)

If R performs a transition λ, different from `throw`, we have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid R) \xrightarrow{\lambda} (\mathbf{new}\,\widetilde{n}')(\bar{P} \mid R') \tag{A.13.13}$$

where (A.13.13) is derived from

$$R \xrightarrow{\lambda'} R' \tag{A.13.14}$$

having $\lambda = (\widetilde{\nu n''})\lambda'$, i.e., label λ is obtained by placing a set of bound names in front of λ', a set that corresponds exactly to the names that λ' extrudes that are restricted names contained in $\widetilde{n}$, hence $\widetilde{n''} = out(\lambda') \cap \widetilde{n}$. Also we have that the resulting restricted name set $\widetilde{n}'$ corresponds to the initial one $\widetilde{n}$ minus the names extruded, hence $(\widetilde{\nu n}' = \widetilde{n} \;/\; \widetilde{n''})$. We also have that $(bn(\lambda') \cap fn(\bar{P}) = \emptyset)$. We are interested only in the transitions λ such that $bn(\lambda) \cap fn((\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R)) = \emptyset$ which gives us that the extruded restricted names must not occur in $(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R)$, hence $n'' \cap fn((\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R)) = \emptyset$, and also that the bound names of λ' also do not occur in $(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R)$, hence

$$bn(\lambda') \cap fn((\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R)) = \emptyset \tag{A.13.15}$$

From (A.13.15) we conclude that

$$bn(\lambda') \cap (fn(\bar{Q}) \;/\; \widetilde{n}) = \emptyset \tag{A.13.16}$$

Considering $\lambda = (\widetilde{\nu n''})\lambda'$ we can be sure that

$$bn(\lambda') \cap \widetilde{n} = \emptyset \tag{A.13.17}$$

otherwise we could not have derived (A.13.13) from (A.13.14).

Finally from (A.13.16) and (A.13.3) we conclude

$$bn(\lambda') \cap fn(Q) = \emptyset \tag{A.13.18}$$

From (A.13.14) and (A.13.18) attending to conditions on the restricted names above we can derive

$$(\mathbf{new}\,\widetilde{n})(\bar{Q} \mid R) \xrightarrow{\lambda} (\mathbf{new}\,\widetilde{n}')(\bar{Q} \mid R')$$

By definition of $\mathcal{R}$ (A.13.1) we conclude

$$((\mathbf{new}\,\widetilde{n}')(\bar{P} \mid R'), (\mathbf{new}\,\widetilde{n}')(\bar{Q} \mid R')) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Transition* $n\rho\texttt{here}$ *due to a synchronization of* $\bar{P}$ *and* R)

We have that

$$(\mathbf{new}\,\widetilde{n})(\bar{P} \mid R) \xrightarrow{n\rho\texttt{here}} (\mathbf{new}\,\widetilde{n}, c)(\bar{P}' \mid R') \tag{A.13.19}$$

where (A.13.19) is either derived from

$$\bar{P} \xrightarrow{(\nu c)\texttt{def}\ s} \bar{P}' \tag{A.13.20}$$

and

$$R \xrightarrow{\overline{(\nu c)n\rho\texttt{def}\ s}} R' \tag{A.13.21}$$

or from

$$\bar{P} \xrightarrow{\overline{(\nu c)n\rho\texttt{def}\ s}} \bar{P}' \tag{A.13.22}$$

and

$$R \xrightarrow{(\nu c)\mathtt{def}\ s} R' \tag{A.13.23}$$

assuming that name c does not occur as a free name of Q to simplify presentation - otherwise we could derive the same transition up to some α-conversion on the resulting processes

$$(\mathbf{new}\ \widetilde{n}, c')(\bar{P}'\{c \leftarrow c'\} \mid R'\{c \leftarrow c'\})$$

using "fresh" name c', in such way ensuring this assumption.

We consider the first case where the derivation occurs from (A.13.20) and (A.13.21), being the proof for the latter case analogous. From (A.13.20) (A.13.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{(\nu c)\mathtt{def}\ s} \bar{Q}' \tag{A.13.24}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.13.25}$$

From (A.13.24) and (A.13.21) we can derive

$$(\mathbf{new}\ \widetilde{n})(\bar{Q} \mid R) \xrightarrow{n\mathtt{phere}} (\mathbf{new}\ \widetilde{n}, c)(\bar{Q}' \mid R')$$

From (A.13.25) and by definition of $\mathcal{R}$ (A.13.1) we conclude

$$((\mathbf{new}\ \widetilde{n}, c)(\bar{P}' \mid R'), (\mathbf{new}\ \widetilde{n}, c)(\bar{Q}' \mid R')) \in \mathcal{R}$$

which completes the proof for this last case.

(*Transition τ due to a synchronization of $\bar{P}$ and R*)

Finally we consider the case when $\bar{P}$ and R synchronize. We have that

$$(\mathbf{new}\ \widetilde{n})(\bar{P} \mid R) \xrightarrow{\tau} (\mathbf{new}\ \widetilde{n}, \widetilde{n}')(\bar{P}' \mid R') \tag{A.13.26}$$

where (A.13.26) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu n}')\lambda} \bar{P}' \tag{A.13.27}$$

and

$$R \xrightarrow{\overline{(\widetilde{\nu n}')\lambda}} R' \tag{A.13.28}$$

assuming that names in $\widetilde{n}'$ and the bound names of λ do not occur as free names of Q to simplify presentation - otherwise we could derive the same transition up to some α-conversion on the label $\lambda\{\widetilde{n}' \leftarrow \widetilde{n}''\}$ and on the resulting processes

$$(\mathbf{new}\ \widetilde{n}, \widetilde{n}'')(\bar{P}'\{\widetilde{n}' \leftarrow \widetilde{n}''\} \mid R'\{\widetilde{n}' \leftarrow \widetilde{n}''\})$$

using "fresh" names $\widetilde{n}''$, in such way ensuring this assumption. Baring in mind that $bn((\widetilde{\nu n}')\lambda) \cap fn(Q) = \emptyset$, from (A.13.27) and (A.13.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{(\widetilde{\nu n}')\lambda} \bar{Q}' \tag{A.13.29}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.13.30}$$

From (A.13.29) and (A.13.28) we can derive

$$(\mathbf{new}\ \widetilde{n})(\bar{Q} \mid R) \xrightarrow{\tau} (\mathbf{new}\ \widetilde{n}, \widetilde{n}')(\bar{Q}' \mid R')$$

From (A.13.30) and by definition of $\mathcal{R}$ (A.13.1) we conclude

$$((\mathbf{new}\ \widetilde{n}, \widetilde{n}')(\bar{P}' \mid R'), (\mathbf{new}\ \widetilde{n}, \widetilde{n}')(\bar{Q}' \mid R')) \in \mathcal{R}$$

which completes the proof for this last case.

Lemma A.14 *Let P, Q be processes such that $P \sim Q$. Then for any process R it is the case that* $\texttt{try}\ P\ \texttt{catch}\ R \sim \texttt{try}\ Q\ \texttt{catch}\ R$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(\texttt{try}\ P\ \texttt{catch}\ R, \texttt{try}\ Q\ \texttt{catch}\ R) \mid P \sim Q \wedge \forall R\} \tag{A.14.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \sim$ and hence $(P', Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.14.1), of the form $\texttt{try}\ \bar{P}\ \texttt{catch}\ R$ and $\texttt{try}\ \bar{Q}\ \texttt{catch}\ R$, respectively, for some processes $R, \bar{P}, \bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.14.2}$$

We must consider two possible transitions of $\texttt{try}\ \bar{P}\ \texttt{catch}\ R$: either $\bar{P}$ triggers a `throw` or some λ different from `throw`.

If $\bar{P}$ triggers a `throw` we have

$$\texttt{try}\ \bar{P}\ \texttt{catch}\ R \xrightarrow{\tau} R \mid \bar{P}' \tag{A.14.3}$$

where (A.14.3) is derived from

$$\bar{P} \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.14.4}$$

From (A.14.4) and (A.14.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{\texttt{throw}} \bar{Q}' \tag{A.14.5}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.14.6}$$

From (A.14.5) we can derive

$$\texttt{try}\ \bar{Q}\ \texttt{catch}\ R \xrightarrow{\tau} R \mid \bar{Q}'$$

From (A.14.6) and considering Lemma A.13 we conclude

$$(R \mid \bar{P}', R \mid \bar{Q}') \in \mathcal{R} \cup \sim$$

which concludes the proof for this case.

If $\bar{P}$ triggers a transition λ different from `throw` we have

$$\texttt{try}\ \bar{P}\ \texttt{catch}\ R \xrightarrow{\lambda} \texttt{try}\ \bar{P}'\ \texttt{catch}\ R \tag{A.14.7}$$

where (A.14.7) is derived from

$$\bar{P} \xrightarrow{\lambda} \bar{P}' \tag{A.14.8}$$

and $\lambda \neq \texttt{throw}$. From (A.14.7) and (A.14.2) we conclude that there exists $\bar{Q}'$ such that

$$\bar{Q} \xrightarrow{\lambda} \bar{Q}' \tag{A.14.9}$$

and

$$\bar{P}' \sim \bar{Q}' \tag{A.14.10}$$

From (A.14.9) we can derive

$$\texttt{try}\ \bar{Q}\ \texttt{catch}\ R \xrightarrow{\lambda} \texttt{try}\ \bar{Q}'\ \texttt{catch}\ R$$

and from (A.14.10) and by definition of $\mathcal{R}$ (A.14.1) we conclude that

$$(\texttt{try}\ \bar{P}'\ \texttt{catch}\ R, \texttt{try}\ \bar{Q}'\ \texttt{catch}\ R) \in \mathcal{R} \cup \sim$$

which completes the proof for this last case.

Lemma A.15 *Let P, Q be processes such that $P \sim Q$. Then for any process R it is the case that* $\texttt{try}\ R\ \texttt{catch}\ P \sim \texttt{try}\ R\ \texttt{catch}\ Q$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(\texttt{try}\ R\ \texttt{catch}\ P, \texttt{try}\ R\ \texttt{catch}\ Q) \mid P \sim Q \wedge \forall R\} \tag{A.15.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P,Q) \in \mathcal{R}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \sim$ and hence $(P', Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.15.1), of the form $\texttt{try}\ R\ \texttt{catch}\ \bar{P}$ and $\texttt{try}\ R\ \texttt{catch}\ \bar{Q}$, respectively, for some processes $R, \bar{P}, \bar{Q}$ such that

$$\bar{P} \sim \bar{Q} \tag{A.15.2}$$

We must consider two possible transitions of $\texttt{try}\ R\ \texttt{catch}\ \bar{P}$: either R triggers a `throw` or some λ different from `throw`.

If R triggers a `throw` we have

$$\texttt{try}\ R\ \texttt{catch}\ \bar{P} \xrightarrow{\tau} \bar{P} \mid R' \tag{A.15.3}$$

where (A.15.3) is derived from

$$R \xrightarrow{\texttt{throw}} R' \tag{A.15.4}$$

From (A.15.4) we can derive

$$\texttt{try}\ R\ \texttt{catch}\ \bar{Q} \xrightarrow{\tau} \bar{Q} \mid R'$$

From (A.15.2) and considering Lemma A.13 we conclude

$$(\bar{P} \mid R', \bar{Q} \mid R') \in \mathcal{R} \cup \sim$$

which concludes the proof for this case.

If R triggers a transition λ different from `throw` we have

$$\texttt{try}\ R\ \texttt{catch}\ \bar{P} \xrightarrow{\lambda} \texttt{try}\ R'\ \texttt{catch}\ \bar{P} \tag{A.15.5}$$

where (A.15.5) is derived from

$$R \xrightarrow{\lambda} R' \tag{A.15.6}$$

and $\lambda \neq \texttt{throw}$. From (A.15.6) we can derive

$$\texttt{try}\ R\ \texttt{catch}\ \bar{Q} \xrightarrow{\lambda} \texttt{try}\ R'\ \texttt{catch}\ \bar{Q}$$

and from (A.1.2) and by definition of $\mathcal{R}$ (A.1.1) we conclude that

$$(\texttt{try}\ R'\ \texttt{catch}\ \bar{P}, \texttt{try}\ R'\ \texttt{catch}\ \bar{Q}) \in \mathcal{R} \cup \sim$$

which completes the proof for this last case.

Lemma A.16 *Let P, Q be processes such that $P \sim Q$. Then it is the case that $!P \sim !Q$.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{P_1 \mid !P_2 \sim Q_1 \mid !Q_2 \mid P_1 \sim Q_1 \wedge P_2 \sim Q_2\} \tag{A.16.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P, Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P, Q) \in \mathcal{R}$ or $(P, Q) \in \sim$.

If $(P, Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \sim$ and hence $(P', Q') \in \mathcal{R} \cup \sim$.

If $(P, Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.16.1), of the form $P_1 \mid !P_2$ and $Q_1 \mid !Q_2$, respectively, for some processes P_1, P_2, Q_1, Q_2 such that

$$P_1 \sim Q_1 \tag{A.16.2}$$

and

$$P_2 \sim Q_2 \tag{A.16.3}$$

We must consider all possible transitions of $P_1 \mid !P_2$: either P_1 triggers a transition, on either a `throw` or on some other label, or one of the copies of $!P_2$ triggers a transition, again either on a `throw` or on some other label, or a synchronization takes place: either one of the copies of $!P_2$ synchronizes with P_1 or two copies of $!P_2$ synchronize.

(*Transition* `throw` *triggered by* P_1)

We have that

$$P_1 \mid !P_2 \xrightarrow{\texttt{throw}} P_1' \tag{A.16.4}$$

where (A.16.4) is derived from

$$P_1 \xrightarrow{\texttt{throw}} P_1' \tag{A.16.5}$$

From (A.16.2) and (A.16.5) we conclude that there exists ${Q_1}'$ such that

$$Q_1 \xrightarrow{\texttt{throw}} Q_1' \tag{A.16.6}$$

and

$${P_1}' \sim {Q_1}' \tag{A.16.7}$$

From (A.16.6) we conclude

$$Q_1 \mid !Q_2 \xrightarrow{\texttt{throw}} Q_1'$$

From (A.16.7) we directly have that

$$(P_1', Q_1') \in \mathcal{R} \cup \sim$$

which concludes the proof for this case.

(*Transition* λ *triggered by* P_1)

We have that

$$P_1 \mid !P_2 \xrightarrow{\lambda} P_1' \mid !P_2 \tag{A.16.8}$$

where (A.16.8) is derived from

$$P_1 \xrightarrow{\lambda} P_1' \tag{A.16.9}$$

From (A.16.2) and (A.16.9) we conclude that there exists ${Q_1}'$ such that

$$Q_1 \xrightarrow{\lambda} Q_1' \tag{A.16.10}$$

and

$${P_1}' \sim {Q_1}' \tag{A.16.11}$$

From (A.16.10) we conclude

$$Q_1 \mid !Q_2 \xrightarrow{\lambda} Q_1' \mid !Q_2$$

Considering (A.16.11) and (A.16.3), by definition of $\mathcal{R}$ (A.16.1) we conclude

$$(P_1' \mid !P_2, Q_1' \mid !Q_2) \in \mathcal{R} \cup \sim$$

which concludes the proof for this case.

(*Transition* `throw` *triggered by one of the copies of* $!P_2$)

We have that

$$P_1 \mid !P_2 \xrightarrow{\texttt{throw}} P_2' \tag{A.16.12}$$

where (A.16.12) is derived from

$$P_2 \mid !P_2 \xrightarrow{\texttt{throw}} P_2' \tag{A.16.13}$$

and (A.16.13) is derived from

$$P_2 \xrightarrow{\texttt{throw}} P_2' \tag{A.16.14}$$

From (A.16.3) and (A.16.14) we conclude that there exists $Q_2{}'$ such that

$$Q_2 \xrightarrow{\texttt{throw}} Q_2' \tag{A.16.15}$$

and

$$P_2{}' \sim Q_2{}' \tag{A.16.16}$$

From (A.16.15) we conclude

$$Q_2 \mid !Q_2 \xrightarrow{\texttt{throw}} Q_2' \tag{A.16.17}$$

and from (A.16.17) we conclude

$$Q_1 \mid !Q_2 \xrightarrow{\texttt{throw}} Q_2'$$

From (A.16.16) we directly have that

$$(P_2', Q_2') \in \mathcal{R} \cup \sim$$

which concludes the proof for this case.

(*Transition* λ *triggered by one of the copies of* P_2)

We have that

$$P_1 \mid !P_2 \xrightarrow{\lambda} P_1 \mid P_2{}' \mid !P_2 \tag{A.16.18}$$

where (A.16.18) is derived from

$$P_2 \mid !P_2 \xrightarrow{\lambda} P_2' \mid !P_2 \tag{A.16.19}$$

and (A.16.19) is derived from

$$P_2 \xrightarrow{\lambda} P_2' \tag{A.16.20}$$

From (A.16.3) and (A.16.20) we conclude that there exists $Q_2{}'$ such that

$$Q_2 \xrightarrow{\lambda} Q_2' \tag{A.16.21}$$

and

$$P_2{}' \sim Q_2{}' \tag{A.16.22}$$

From (A.16.21) we conclude

$$Q_2 \mid !Q_2 \xrightarrow{\lambda} Q_2' \mid !Q_2 \tag{A.16.23}$$

and from (A.16.23) we conclude

$$Q_1 \mid !Q_2 \xrightarrow{\lambda} Q_1 \mid Q_2' \mid !Q_2$$

From (A.16.22) and (A.16.2), building on Lemma A.13 we conclude

$$P_1 \mid P_2' \sim Q_1 \mid Q_2' \tag{A.16.24}$$

From (A.16.24) and (A.16.3), by definition of $\mathcal{R}$ (A.16.1) we conclude

$$(P_1 \mid P_2' \mid !P_2, Q_1 \mid Q_2' \mid !Q_2) \in \mathcal{R} \cup \sim$$

which concludes the proof for this case.

(*Synchronization between* P_1 *and one of the copies of* $!P_2$)

We have that

$$P_1 \mid !P_2 \xrightarrow{\tau} (\mathbf{new}\, \widetilde{n})(P_1' \mid P_2{}') \mid !P_2 \tag{A.16.25}$$

where (A.16.25) is derived from

$$P_1 \mid P_2 \mid !P_2 \xrightarrow{\tau} (\mathbf{new}\, \widetilde{n})(P_1' \mid P_2{}') \mid !P_2 \tag{A.16.26}$$

and (A.16.26) is derived from

$$P_1 \mid P_2 \xrightarrow{\tau} (\mathbf{new}\, \widetilde{n})(P_1' \mid P_2{}') \tag{A.16.27}$$

and (A.16.27) is derived from

$$P_1 \xrightarrow{(\widetilde{\nu n})\lambda} P_1' \tag{A.16.28}$$

and

$$P_2 \xrightarrow{\overline{(\widetilde{\nu n})\lambda}} P_2' \tag{A.16.29}$$

From (A.16.2) and (A.16.28) we conclude that there exists $Q_1{}'$ such that

$$Q_1 \xrightarrow{(\widetilde{\nu n})\lambda} Q_1' \tag{A.16.30}$$

and

$$P_1{}' \sim Q_1{}' \tag{A.16.31}$$

From (A.16.3) and (A.16.29) we conclude that there exists $Q_2{}'$ such that

$$Q_2 \xrightarrow{\overline{(\widetilde{\nu n})\lambda}} Q_2' \tag{A.16.32}$$

and

$$P_2{}' \sim Q_2{}' \tag{A.16.33}$$

From (A.16.30) and (A.16.32) we conclude

$$Q_1 \mid Q_2 \xrightarrow{\tau} (\mathbf{new}\, \widetilde{n})(Q_1' \mid Q_2') \tag{A.16.34}$$

and from (A.16.34) we conclude

$$Q_1 \mid Q_2 \mid !Q_2 \xrightarrow{\tau} (\mathbf{new}\, \widetilde{n})(Q_1' \mid Q_2') \mid !Q_2 \tag{A.16.35}$$

and finally from (A.16.35) we conclude

$$Q_1 \mid !Q_2 \xrightarrow{\tau} (\mathbf{new}\, \widetilde{n})(Q_1' \mid Q_2') \mid !Q_2$$

From (A.16.31) and (A.16.33), building on Lemma A.13 we conclude

$$P_1' \mid P_2' \sim Q_1' \mid Q_2' \tag{A.16.36}$$

and from (A.16.36) and considering Lemma A.11 we conclude

$$(\mathbf{new}\, \widetilde{n})(P_1' \mid P_2') \sim (\mathbf{new}\, \widetilde{n})(Q_1' \mid Q_2') \tag{A.16.37}$$

From (A.16.37) and (A.16.3), by definition of $\mathcal{R}$ (A.16.1) we conclude

$$((\mathbf{new}\, \widetilde{n})(P_1' \mid P_2') \mid !P_2, (\mathbf{new}\, \widetilde{n})(Q_1' \mid Q_2') \mid !Q_2) \in \mathcal{R} \cup \sim$$

which concludes the proof for this case.

(*Synchronization between two copies of* $!P_2$)
We have that

$$P_1 \mid !P_2 \xrightarrow{\tau} P_1 \mid (\mathtt{new}\ \widetilde{n})(P_2' \mid P_2'') \mid !P_2 \tag{A.16.38}$$

where (A.16.38) is derived from

$$P_1 \mid P_2 \mid P_2 \mid !P_2 \xrightarrow{\tau} P_1 \mid (\mathtt{new}\ \widetilde{n})(P_2' \mid P_2'') \mid !P_2 \tag{A.16.39}$$

and (A.16.39) is derived from

$$P_2 \mid P_2 \xrightarrow{\tau} (\mathtt{new}\ \widetilde{n})(P_2' \mid P_2'') \tag{A.16.40}$$

and (A.16.40) is derived from

$$P_2 \xrightarrow{(\widetilde{\nu n})\lambda} P_2' \tag{A.16.41}$$

and

$$P_2 \xrightarrow{\overline{(\widetilde{\nu n})\lambda}} P_2'' \tag{A.16.42}$$

From (A.16.3) and (A.16.41) we conclude that there exists $Q_2{}'$ such that

$$Q_2 \xrightarrow{(\widetilde{\nu n})\lambda} Q_2' \tag{A.16.43}$$

and

$$P_2{}' \sim Q_2{}' \tag{A.16.44}$$

From (A.16.3) and (A.16.42) we conclude that there exists $Q_2{}''$ such that

$$Q_2 \xrightarrow{\overline{(\widetilde{\nu n})\lambda}} Q_2'' \tag{A.16.45}$$

and

$$P_2{}'' \sim Q_2{}'' \tag{A.16.46}$$

From (A.16.43) and (A.16.45) we conclude

$$Q_2 \mid Q_2 \xrightarrow{\tau} (\mathtt{new}\ \widetilde{n})(Q_2' \mid Q_2'') \tag{A.16.47}$$

and from (A.16.47) we conclude

$$Q_1 \mid Q_2 \mid Q_2 \mid !Q_2 \xrightarrow{\tau} Q_1 \mid (\mathtt{new}\ \widetilde{n})(Q_2' \mid Q_2'') \mid !Q_2 \tag{A.16.48}$$

and finally from (A.16.48) we conclude

$$Q_1 \mid !Q_2 \xrightarrow{\tau} Q_1 \mid (\mathtt{new}\ \widetilde{n})(Q_2' \mid Q_2'') \mid !Q_2$$

From (A.16.44) and (A.16.46), building on Lemma A.13 we conclude

$$P_2' \mid P_2'' \sim Q_2' \mid Q_2'' \tag{A.16.49}$$

and from (A.16.49) and considering Lemma A.11 we conclude

$$(\mathtt{new}\ \widetilde{n})(P_2' \mid P_2'') \sim (\mathtt{new}\ \widetilde{n})(Q_2' \mid Q_2'') \tag{A.16.50}$$

From (A.16.50) and (A.16.2), building on Lemma A.13 we conclude

$$P_1 \mid (\mathtt{new}\ \widetilde{n})(P_2' \mid P_2'') \sim Q_1 \mid (\mathtt{new}\ \widetilde{n})(Q_2' \mid Q_2'') \tag{A.16.51}$$

From (A.16.51) and (A.16.3), by definition of $\mathcal{R}$ (A.16.1) we conclude

$$(P_1 \mid (\mathtt{new}\ \widetilde{n})(P_2' \mid P_2'') \mid !P_2, Q_1 \mid (\mathtt{new}\ \widetilde{n})(Q_2' \mid Q_2'') \mid !Q_2) \in \mathcal{R} \cup \sim$$

which concludes the proof for this case.

Proof of auxiliar results to Proposition 5.4

Lemma A.17 *We have* $(\mathtt{new}\,\widetilde{m})(n \blacktriangleright [P]) \sim n \blacktriangleright [(\mathtt{new}\,\widetilde{m})(P)]$ *for any process P and names* $n, \widetilde{m}$ *such that* $n \notin \widetilde{m}$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{((\mathtt{new}\,\widetilde{m})(n \blacktriangleright [P]), n \blacktriangleright [(\mathtt{new}\,\widetilde{m})(P)]) \mid \forall P, n, \widetilde{m} \,.\, n \notin \widetilde{m}\} \cup \{(P,P)\} \tag{A.17.1}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R}$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R}$.

We must consider two different cases: either

$$(P,Q) \in \{((\mathtt{new}\,\widetilde{m})(n \blacktriangleright [P]), n \blacktriangleright [(\mathtt{new}\,\widetilde{m})(P)]) \mid \forall P, n, \widetilde{m} \,.\, n \notin \widetilde{m}\}$$

or $(P,Q) \in \{(P,P)\}$.

If $(P,Q) \in \{(P,P)\}$ we directly have that $(P',P') \in \{(P,P)\}$ and hence $(P',P') \in \mathcal{R}$.

Otherwise we have that P and Q are of the form $(\mathtt{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}])$ and $n \blacktriangleright [(\mathtt{new}\,\widetilde{m})(\bar{P})]$, respectively, for some process $\bar{P}$ and names $n, \widetilde{m}$ such that $n \notin \widetilde{m}$.

We have that all possible transitions of $(\mathtt{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}])$ are triggered by $\bar{P}$ so we have that

$$(\mathtt{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}]) \xrightarrow{\lambda} (\mathtt{new}\,\widetilde{m}')\bar{P}' \tag{A.17.2}$$

where (A.17.2) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{\lambda'} \bar{P}' \tag{A.17.3}$$

having $\lambda = (\widetilde{\nu m''})\lambda'$, i.e., label λ is obtained by placing a set of bound names in front of λ', a set that corresponds exactly to the names that λ' extrudes that are restricted names contained in $\widetilde{m}$, hence $\widetilde{m}'' = out(\lambda') \cap \widetilde{m}$. Also we have that the resulting restricted name set $\widetilde{m}'$ corresponds to the initial one $\widetilde{m}$ minus the names extruded, hence $(\widetilde{\nu m}' = \widetilde{m} \,/\, \widetilde{m}'')$.

We have that (A.17.3) is derived from one of either seven distinct cases: either from a `throw`, or from a $(\widetilde{\nu o})m\rho\lambda''$, where $\lambda'' \neq \mathtt{here}$, or from $n \blacktriangleright \mathtt{here}$, or from $(\widetilde{\nu o})\lambda''^{\leftarrow}$, or from $(\widetilde{\nu o})\lambda''^{\downarrow}$ or $(\widetilde{\nu o})\lambda''^{\uparrow}$ or finally from $(\nu c)\mathtt{def}\ s$.

We must also consider all possible transitions of $n \blacktriangleright [(\mathtt{new}\,\widetilde{m})\bar{P}]$ which proof we omit given that it follows similar lines.

(*Case* `throw`)

We have that

$$(\mathtt{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}]) \xrightarrow{\mathtt{throw}} (\mathtt{new}\,\widetilde{m})\bar{P}' \tag{A.17.4}$$

where (A.17.4) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{\mathtt{throw}} \bar{P}' \tag{A.17.5}$$

and (A.17.5) is derived from

$$\bar{P} \xrightarrow{\mathtt{throw}} \bar{P}' \tag{A.17.6}$$

From (A.17.6) we can derive

$$(\mathtt{new}\,\widetilde{m})\bar{P} \xrightarrow{\mathtt{throw}} (\mathtt{new}\,\widetilde{m})\bar{P}' \tag{A.17.7}$$

From (A.17.7) we derive

$$n \blacktriangleright [(\mathbf{new}\ \widetilde{m})\bar{P}] \xrightarrow{\mathtt{throw}} (\mathbf{new}\ \widetilde{m})\bar{P}'$$

By definition of $\mathcal{R}$ (A.17.1) we have that

$$((\mathbf{new}\ \widetilde{m})\bar{P}', (\mathbf{new}\ \widetilde{m})\bar{P}') \in \mathcal{R}$$

which completes the proof for this case.

(*Case* $(\widetilde{\nu o})m\rho\lambda''$)

We have that

$$(\mathbf{new}\ \widetilde{m})(n \blacktriangleright [\bar{P}]) \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})m\rho\lambda''} (\mathbf{new}\ \widetilde{m'})(n \blacktriangleright \left[\bar{P}'\right]) \tag{A.17.8}$$

where $out(\lambda'') \cap \widetilde{m} = \widetilde{m''}$ and $\widetilde{m'} = \widetilde{m} \setminus \widetilde{m''}$.

(A.17.8) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})m\rho\lambda''} n \blacktriangleright \left[\bar{P}'\right] \tag{A.17.9}$$

and (A.17.9) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})m\rho\lambda''} \bar{P}' \tag{A.17.10}$$

and $\lambda'' \neq \mathtt{here}$. From (A.17.10) and recalling that $out(\lambda') \cap \widetilde{m} = \widetilde{m''}$ and $\widetilde{m'} = \widetilde{m} \setminus \widetilde{m''}$ we can derive

$$(\mathbf{new}\ \widetilde{m})\bar{P} \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})m\rho\lambda''} (\mathbf{new}\ \widetilde{m'})\bar{P}' \tag{A.17.11}$$

From (A.17.11) and noting that $\lambda'' \neq \mathtt{here}$ we derive

$$n \blacktriangleright [(\mathbf{new}\ \widetilde{m})\bar{P}] \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})m\rho\lambda''} n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m'})\bar{P}'\right]$$

By definition of $\mathcal{R}$ (A.17.1) we have that

$$((\mathbf{new}\ \widetilde{m'})(n \blacktriangleright \left[\bar{P}'\right]), n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m'})\bar{P}'\right]) \in \mathcal{R}$$

which completes the proof for this case.

(*Case* $n \blacktriangleright \mathtt{here}$)

We have that

$$(\mathbf{new}\ \widetilde{m})(n \blacktriangleright [\bar{P}]) \xrightarrow{\tau} (\mathbf{new}\ \widetilde{m})(n \blacktriangleright \left[\bar{P}'\right]) \tag{A.17.12}$$

where (A.17.12) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{\tau} n \blacktriangleright \left[\bar{P}'\right] \tag{A.17.13}$$

and (A.17.13) is derived from

$$\bar{P} \xrightarrow{n \blacktriangleright \mathtt{here}} \bar{P}' \tag{A.17.14}$$

From (A.17.14) and recalling that $n \notin \widetilde{m}$ we can derive

$$(\mathbf{new}\ \widetilde{m})\bar{P} \xrightarrow{n \blacktriangleright \mathtt{here}} (\mathbf{new}\ \widetilde{m})\bar{P}' \tag{A.17.15}$$

From (A.17.15) we derive

$$n \blacktriangleright [(\mathbf{new}\ \widetilde{m})\bar{P}] \xrightarrow{\tau} n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})\bar{P}'\right]$$

By definition of $\mathcal{R}$ (A.17.1) we have that

$$((\mathbf{new}\ \widetilde{m})(n \blacktriangleright \left[\bar{P}'\right]), n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})\bar{P}'\right]) \in \mathcal{R}$$

which completes the proof for this case.

(*Case* $(\widetilde{\nu o})\lambda''^{\leftarrow}$)

We have that

$$(\mathbf{new}\ \widetilde{m})(n \blacktriangleright [\bar{P}]) \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})n\blacktriangleright\cdot\lambda''^{\leftarrow}} (\mathbf{new}\ \widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right]) \tag{A.17.16}$$

where $out(\lambda'') \cap \widetilde{m} = \widetilde{m''}$ and $\widetilde{m}' = \widetilde{m} \setminus \widetilde{m''}$.

(A.17.16) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda''^{\leftarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.17.17}$$

and (A.17.17) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda''^{\leftarrow}} \bar{P}' \tag{A.17.18}$$

From (A.17.18) and $out(\lambda'') \cap \widetilde{m} = \widetilde{m''}$ and $\widetilde{m}' = \widetilde{m} \setminus \widetilde{m''}$ we derive

$$(\mathbf{new}\ \widetilde{m})\bar{P} \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})\lambda''^{\leftarrow}} (\mathbf{new}\ \widetilde{m}')\bar{P}' \tag{A.17.19}$$

From (A.17.19) we can derive

$$n \blacktriangleright [(\mathbf{new}\ \widetilde{m})\bar{P}] \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})n\blacktriangleright\cdot\lambda''^{\leftarrow}} n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}')\bar{P}'\right]$$

By definition of $\mathcal{R}$ (A.17.1) we have that

$$((\mathbf{new}\ \widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right]), n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}')\bar{P}'\right]) \in \mathcal{R}$$

which completes the proof for this case.

(*Case* $(\widetilde{\nu o})\lambda''^{\downarrow}$)

We have that

$$(\mathbf{new}\ \widetilde{m})(n \blacktriangleright [\bar{P}]) \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})n\blacktriangleright\cdot\lambda''^{\downarrow}} (\mathbf{new}\ \widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right]) \tag{A.17.20}$$

where $out(\lambda'') \cap \widetilde{m} = \widetilde{m''}$ and $\widetilde{m}' = \widetilde{m} \setminus \widetilde{m''}$.

(A.17.20) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda''^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.17.21}$$

and (A.17.21) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda''^{\downarrow}} \bar{P}' \tag{A.17.22}$$

From (A.17.22) and $out(\lambda'') \cap \widetilde{m} = \widetilde{m''}$ and $\widetilde{m}' = \widetilde{m} \setminus \widetilde{m''}$ we derive

$$(\mathbf{new}\ \widetilde{m})\bar{P} \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})\lambda''^{\downarrow}} (\mathbf{new}\ \widetilde{m}')\bar{P}' \tag{A.17.23}$$

From (A.17.23) we can derive

$$n \blacktriangleright [(\mathbf{new}\ \widetilde{m})\bar{P}] \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})n\blacktriangleright\cdot\lambda''^{\downarrow}} n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}')\bar{P}'\right]$$

By definition of $\mathcal{R}$ (A.17.1) we have that

$$((\mathbf{new}\ \widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right]), n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}')\bar{P}'\right]) \in \mathcal{R}$$

which completes the proof for this case.

(*Case* $(\widetilde{\nu o})\lambda''^{\uparrow}$)
We have that

$$(\mathtt{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}]) \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})\lambda''^{\downarrow}} (\mathtt{new}\,\widetilde{m'})(n \blacktriangleright \left[\bar{P'}\right]) \tag{A.17.24}$$

where $out(\lambda'') \cap \widetilde{m} = \widetilde{m''}$ and $\widetilde{m'} = \widetilde{m} \setminus \widetilde{m''}$.
(A.17.24) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})\lambda''^{\downarrow}} n \blacktriangleright \left[\bar{P'}\right] \tag{A.17.25}$$

and (A.17.25) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda''^{\uparrow}} \bar{P'} \tag{A.17.26}$$

From (A.17.26) and $out(\lambda'') \cap \widetilde{m} = \widetilde{m''}$ and $\widetilde{m'} = \widetilde{m} \setminus \widetilde{m''}$ we derive

$$(\mathtt{new}\,\widetilde{m})\bar{P} \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})\lambda''^{\uparrow}} (\mathtt{new}\,\widetilde{m'})\bar{P'} \tag{A.17.27}$$

From (A.17.27) we can derive

$$n \blacktriangleright [(\mathtt{new}\,\widetilde{m})\bar{P}] \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})\lambda''^{\uparrow}} n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m'})\bar{P'}\right]$$

By definition of $\mathcal{R}$ (A.17.1) we have that

$$((\mathtt{new}\,\widetilde{m'})(n \blacktriangleright \left[\bar{P'}\right]), n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m'})\bar{P'}\right]) \in \mathcal{R}$$

which completes the proof for this case.
(*Case* $(\nu c)\mathtt{def}\ s$)
We have that

$$(\mathtt{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}]) \xrightarrow{(\nu c)n \blacktriangleright \mathtt{def}\ s} (\mathtt{new}\,\widetilde{m})(n \blacktriangleright \left[\bar{P'}\right]) \tag{A.17.28}$$

where (A.17.28) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\nu c)n \blacktriangleright \mathtt{def}\ s} n \blacktriangleright \left[\bar{P'}\right] \tag{A.17.29}$$

and $\{c, n, s\} \cap \widetilde{m} = \emptyset$.
(A.17.29) is derived from

$$\bar{P} \xrightarrow{(\nu c)\mathtt{def}\ s} \bar{P'} \tag{A.17.30}$$

From (A.17.30) and $\{c, n, s\} \cap \widetilde{m} = \emptyset$ we derive

$$(\mathtt{new}\,\widetilde{m})\bar{P} \xrightarrow{(\nu c)\mathtt{def}\ s} (\mathtt{new}\,\widetilde{m})\bar{P'} \tag{A.17.31}$$

From (A.17.31) we can derive

$$n \blacktriangleright [(\mathtt{new}\,\widetilde{m})\bar{P}] \xrightarrow{(\nu c)n \blacktriangleright \mathtt{def}\ s} n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m})\bar{P'}\right]$$

By definition of $\mathcal{R}$ (A.17.1) we have that

$$((\mathtt{new}\,\widetilde{m})(n \blacktriangleright \left[\bar{P'}\right]), n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m})\bar{P'}\right]) \in \mathcal{R}$$

which completes the proof for this last case.

Lemma A.18 *We have* $(\mathtt{new}\,\widetilde{m})(n \blacktriangleleft [P]) \sim n \blacktriangleleft [(\mathtt{new}\,\widetilde{m})(P)]$ *for any process P and names* $n, \widetilde{m}$ *such that* $n \notin \widetilde{m}$.

Proof. Analogous to that of Lemma A.17.

Lemma A.19 *We have $n \blacktriangleright [P] \mid n \blacktriangleright [Q] \sim n \blacktriangleright [P \mid Q]$, for any name n and processes P, Q.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{((\mathbf{new}\, \widetilde{m})(n \blacktriangleright [P] \mid n \blacktriangleright [Q]), n \blacktriangleright [(\mathbf{new}\, \widetilde{m})(P \mid Q)]) \mid \forall P, Q, n, \widetilde{m} \,.\, n \notin \widetilde{m}\} \tag{A.19.1}$$

We show that $\mathcal{R} \cup \sim$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P, Q) \in \mathcal{R} \cup \sim$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R} \cup \sim$.

We must consider two different cases: either $(P, Q) \in \mathcal{R}$ or $(P, Q) \in \sim$.

If $(P, Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \sim$ and hence $(P', Q') \in \mathcal{R} \cup \sim$.

If $(P, Q) \in \mathcal{R}$ we have that P and Q are, by definition of $\mathcal{R}$ (A.19.1) of the form $(\mathbf{new}\, \widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right])$ and $(\mathbf{new}\, \widetilde{m})(n \blacktriangleright \left[\bar{P} \mid \bar{Q}\right])$, respectively, for some name n and set of names $\widetilde{m}$ and processes $\bar{P}, \bar{Q}$ such that $n \notin \widetilde{m}$.

We must consider three distinct cases for the derivation of the transition of $(\mathbf{new}\, \widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right])$: it is either triggered by $\bar{P}$ or by $\bar{Q}$ or by a synchronization of $\bar{P}$ and $\bar{Q}$.

(**Transition triggered by $\bar{P}$**)

We have that the transition λ can be derived from a `throw` or from a $(\widetilde{\nu o})m\rho\lambda'$, where $\lambda' \neq \texttt{here}$, or from $n \blacktriangleright \texttt{here}$, or from $(\widetilde{\nu o})\lambda'^{\leftarrow}$, or from $(\widetilde{\nu o})\lambda'^{\downarrow}$ or from $(\widetilde{\nu o})\lambda'^{\uparrow}$ or finally from $(\nu c)\texttt{def}\ s$.

(*Case* `throw`)

We have that

$$(\mathbf{new}\, \widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\texttt{throw}} (\mathbf{new}\, \widetilde{m})\bar{P}' \tag{A.19.2}$$

where (A.19.2) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.19.3}$$

and (A.19.3) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.19.4}$$

and (A.19.4) is derived from

$$\bar{P} \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.19.5}$$

From (A.19.5) we can derive

$$\bar{P} \mid \bar{Q} \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.19.6}$$

and from (A.19.7) we conclude

$$(\mathbf{new}\, \widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{\texttt{throw}} (\mathbf{new}\, \widetilde{m})(\bar{P}') \tag{A.19.7}$$

From (A.19.7) we derive

$$n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\texttt{throw}} (\mathbf{new}\,\widetilde{m})\bar{P}'$$

We directly have that

$$((\mathbf{new}\,\widetilde{m})\bar{P}', (\mathbf{new}\,\widetilde{m})\bar{P}') \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $(\widetilde{\nu o})m\rho\lambda'$)

We have that

$$(\mathbf{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})m\rho\lambda'} (\mathbf{new}\,\widetilde{m'})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]) \tag{A.19.8}$$

where $\widetilde{m}''$ corresponds to the set of names extruded by λ' contained in $\widetilde{m}$ and $\widetilde{m}'$ the remaining set of restricted names, being that (A.19.8) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})m\rho\lambda'} n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right] \tag{A.19.9}$$

and $(n((\widetilde{\nu o})m\rho\lambda') \,/\, \widetilde{m}'') \cap \widetilde{m} = \emptyset$.

(A.19.9) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})m\rho\lambda'} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.10}$$

and $bn((\widetilde{\nu o})m\rho\lambda') \cap fn(n \blacktriangleright \left[\bar{Q}\right]) = \emptyset$.

(A.19.10) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})m\rho\lambda'} \bar{P}' \tag{A.19.11}$$

and it is the case that $\lambda' \neq \texttt{here}$. From (A.19.11) and from the fact that $bn((\widetilde{\nu o})m\rho\lambda') \cap fn(Q) = \emptyset$ we can derive

$$\bar{P} \mid \bar{Q} \xrightarrow{(\widetilde{\nu o})m\rho\lambda'} \bar{P}' \mid \bar{Q} \tag{A.19.12}$$

We have that $out(\lambda') \cap \widetilde{m} = \widetilde{m}''$ and $\widetilde{m}' = \widetilde{m} \,/\, \widetilde{m}''$ hence from (A.19.12) we can derive

$$(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})m\rho\lambda'} (\mathbf{new}\,\widetilde{m'})(\bar{P}' \mid \bar{Q}) \tag{A.19.13}$$

From (A.19.13) considering that $\lambda' \neq \texttt{here}$ we derive

$$n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{(\widetilde{\nu m''})(\widetilde{\nu o})m\rho\lambda'} n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m'})(\bar{P}' \mid \bar{Q})\right]$$

By definition of $\mathcal{R}$ (A.19.1) we have that

$$((\mathbf{new}\,\widetilde{m'})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]), n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m'})(\bar{P}' \mid \bar{Q})\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $n \blacktriangleright \texttt{here}$)

We have that

$$(\mathbf{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\tau} (\mathbf{new}\,\widetilde{m})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]) \tag{A.19.14}$$

where (A.19.14) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right] \tag{A.19.15}$$

and (A.19.15) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{\tau} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.16}$$

and (A.19.16) is derived from

$$\bar{P} \xrightarrow{n\blacktriangleright\mathtt{here}} \bar{P}' \tag{A.19.17}$$

From (A.19.17) we can derive

$$\bar{P} \mid \bar{Q} \xrightarrow{n\blacktriangleright\mathtt{here}} \bar{P}' \mid \bar{Q} \tag{A.19.18}$$

From (A.19.18) noting that $n \notin \widetilde{m}$ we can derive

$$(\mathtt{new}\,\widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{n\blacktriangleright\mathtt{here}} (\mathtt{new}\,\widetilde{m})(\bar{P}' \mid \bar{Q}) \tag{A.19.19}$$

From (A.19.19) we conclude

$$n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\tau} n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m})(\bar{P}' \mid \bar{Q})\right]$$

By definition of $\mathcal{R}$ (A.19.1) we have that

$$((\mathtt{new}\,\widetilde{m})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]), n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m})(\bar{P}' \mid \bar{Q})\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $(\widetilde{\nu o})\lambda'^{\leftarrow}$)

We have that

$$(\mathtt{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{(\widetilde{\nu m}'')(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\leftarrow}} (\mathtt{new}\,\widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]) \tag{A.19.20}$$

where $\widetilde{m}''$ corresponds to the set of names extruded by λ' contained in $\widetilde{m}$ and $\widetilde{m}'$ the remaining set of restricted names, being that (A.19.20) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\leftarrow}} n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right] \tag{A.19.21}$$

and $(n((\widetilde{\nu o})n \blacktriangleright \cdot\lambda'^{\leftarrow}) \,/\, \widetilde{m}'') \cap \widetilde{m} = \emptyset$.

(A.19.21) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\leftarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.22}$$

and $bn((\widetilde{\nu o})n \blacktriangleright \cdot\lambda'^{\leftarrow}) \cap fn(n \blacktriangleright \left[\bar{Q}\right]) = \emptyset$.

(A.19.22) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda'^{\leftarrow}} \bar{P}' \tag{A.19.23}$$

From (A.19.23) and from the fact that $bn((\widetilde{\nu o})\lambda'^{\leftarrow}) \cap fn(Q) = \emptyset$ we can derive

$$\bar{P} \mid \bar{Q} \xrightarrow{(\widetilde{\nu o})\lambda'^{\leftarrow}} \bar{P}' \mid \bar{Q} \tag{A.19.24}$$

Considering $(n((\widetilde{\nu o}) \leftarrow \lambda') \,/\, \widetilde{m}'') \cap \widetilde{m} = \emptyset$ and $out(\lambda') \cap \widetilde{m} = \widetilde{m}''$ and $\widetilde{m}' = \widetilde{m} \,/\, \widetilde{m}''$, from (A.19.24) we can derive

$$(\mathtt{new}\,\widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{(\widetilde{\nu m}'')(\widetilde{\nu o})\lambda'^{\leftarrow}} (\mathtt{new}\,\widetilde{m}')(\bar{P}' \mid \bar{Q}) \tag{A.19.25}$$

From (A.19.25) we conclude

$$n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{(\widetilde{\nu m}'')(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\leftarrow}} n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m}')(\bar{P}' \mid \bar{Q})\right]$$

By definition of $\mathcal{R}$ (A.19.1) we have that

$$((\mathtt{new}\,\widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]), n \blacktriangleright \left[(\mathtt{new}\,\widetilde{m}')(\bar{P}' \mid \bar{Q})\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $(\widetilde{\nu o})\lambda'^{\downarrow}$)

We have that

$$(\mathbf{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{(\widetilde{\nu m}'')(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\downarrow}} (\mathbf{new}\,\widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]) \tag{A.19.26}$$

where $\widetilde{m}''$ corresponds to the set of names extruded by λ' contained in $\widetilde{m}$ and $\widetilde{m}'$ the remaining set of restricted names, being that (A.19.26) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right] \tag{A.19.27}$$

and $(n((\widetilde{\nu o})n \blacktriangleright \cdot\lambda'^{\downarrow}) \,/\, \widetilde{m}'') \cap \widetilde{m} = \emptyset$.

(A.19.27) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.28}$$

and $bn((\widetilde{\nu o})n \blacktriangleright \cdot\lambda'^{\downarrow}) \cap fn(n \blacktriangleright \left[\bar{Q}\right]) = \emptyset$.

(A.19.28) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} \bar{P}' \tag{A.19.29}$$

From (A.19.29) and from the fact that $bn((\widetilde{\nu o})\lambda'^{\downarrow}) \cap fn(Q) = \emptyset$ we can derive

$$\bar{P} \mid \bar{Q} \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} \bar{P}' \mid \bar{Q} \tag{A.19.30}$$

Considering $(n((\widetilde{\nu o}) \downarrow \lambda') \,/\, \widetilde{m}'') \cap \widetilde{m} = \emptyset$ and $out(\lambda') \cap \widetilde{m} = \widetilde{m}''$ and $\widetilde{m}' = \widetilde{m} \,/\, \widetilde{m}''$, from (A.19.30) we can derive

$$(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{(\widetilde{\nu m}'')(\widetilde{\nu o})\lambda'^{\downarrow}} (\mathbf{new}\,\widetilde{m}')(\bar{P}' \mid \bar{Q}) \tag{A.19.31}$$

From (A.19.31) we conclude

$$n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{(\widetilde{\nu m}'')(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\downarrow}} n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m}')(\bar{P}' \mid \bar{Q})\right]$$

By definition of $\mathcal{R}$ (A.19.1) we have that

$$((\mathbf{new}\,\widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]), n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m}')(\bar{P}' \mid \bar{Q})\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $(\widetilde{\nu o})\lambda'^{\uparrow}$)

We have that

$$(\mathbf{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{(\widetilde{\nu m}'')(\widetilde{\nu o})\lambda'^{\downarrow}} (\mathbf{new}\,\widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]) \tag{A.19.32}$$

where $\widetilde{m}''$ corresponds to the set of names extruded by λ' contained in $\widetilde{m}$ and $\widetilde{m}'$ the remaining set of restricted names, being that (A.19.32) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right] \tag{A.19.33}$$

and $(n((\widetilde{\nu o})\lambda'^{\downarrow}) \,/\, \widetilde{m}'') \cap \widetilde{m} = \emptyset$.

(A.19.33) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.34}$$

and $bn((\widetilde{\nu o})\lambda'^{\downarrow}) \cap fn(n \blacktriangleright \left[\bar{Q}\right]) = \emptyset$.

(A.19.34) is derived from

$$\bar{P} \stackrel{(\widetilde{\nu o})\lambda'^{\uparrow}}{\longrightarrow} \bar{P}' \tag{A.19.35}$$

From (A.19.35) and from the fact that $bn((\widetilde{\nu o})\lambda'^{\uparrow}) \cap fn(Q) = \emptyset$ we can derive

$$\bar{P} \mid \bar{Q} \stackrel{(\widetilde{\nu o})\lambda'^{\uparrow}}{\longrightarrow} \bar{P}' \mid \bar{Q} \tag{A.19.36}$$

Considering $(n((\widetilde{\nu o}) \uparrow \lambda') \,/\, \widetilde{m}'') \cap \widetilde{m} = \emptyset$ and $out(\lambda') \cap \widetilde{m} = \widetilde{m}''$ and $\widetilde{m}' = \widetilde{m} \,/\, \widetilde{m}''$, from (A.19.36) we can derive

$$(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q}) \stackrel{(\widetilde{\nu m}'')(\widetilde{\nu o})\lambda'^{\uparrow}}{\longrightarrow} (\mathbf{new}\ \widetilde{m}')(\bar{P}' \mid \bar{Q}) \tag{A.19.37}$$

From (A.19.37) we conclude

$$n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q})\right] \stackrel{(\widetilde{\nu m}'')(\widetilde{\nu o})\lambda'^{\downarrow}}{\longrightarrow} n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}')(\bar{P}' \mid \bar{Q})\right]$$

By definition of $\mathcal{R}$ (A.19.1) we have that

$$((\mathbf{new}\ \widetilde{m}')(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]), n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}')(\bar{P}' \mid \bar{Q})\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case* $(\nu c)\mathtt{def}\ s$)

We have that

$$(\mathbf{new}\ \widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \stackrel{(\nu c)n \blacktriangleright \mathtt{def}\ s}{\longrightarrow} (\mathbf{new}\ \widetilde{m})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]) \tag{A.19.38}$$

where (A.19.38) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \stackrel{(\nu c)n \blacktriangleright \mathtt{def}\ s}{\longrightarrow} n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right] \tag{A.19.39}$$

and $n((\nu c)n \blacktriangleright \mathtt{def}\ s) \cap \widetilde{m} = \emptyset$.

(A.19.39) is derived from

$$n \blacktriangleright [\bar{P}] \stackrel{(\nu c)n \blacktriangleright \mathtt{def}\ s}{\longrightarrow} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.40}$$

and $bn((\nu c)n \blacktriangleright \mathtt{def}\ s) \cap fn(n \blacktriangleright \left[\bar{Q}\right]) = \emptyset$.

(A.19.40) is derived from

$$\bar{P} \stackrel{(\nu c)\mathtt{def}\ s}{\longrightarrow} \bar{P}' \tag{A.19.41}$$

From (A.19.41) and from the fact that $bn((\nu c)\mathtt{def}\ s) \cap fn(Q) = \emptyset$ we can derive

$$\bar{P} \mid \bar{Q} \stackrel{(\nu c)\mathtt{def}\ s}{\longrightarrow} \bar{P}' \mid \bar{Q} \tag{A.19.42}$$

Considering $(n((\nu c)\mathtt{def}\ s) \cap \widetilde{m} = \emptyset$, from (A.19.42) we can derive

$$(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q}) \stackrel{(\nu c)\mathtt{def}\ s}{\longrightarrow} (\mathbf{new}\ \widetilde{m})(\bar{P}' \mid \bar{Q}) \tag{A.19.43}$$

From (A.19.43) we conclude

$$n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q})\right] \stackrel{(\nu c)n \blacktriangleright \mathtt{def}\ s}{\longrightarrow} n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})(\bar{P}' \mid \bar{Q})\right]$$

By definition of $\mathcal{R}$ (A.19.1) we have that

$$((\mathbf{new}\ \widetilde{m})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}\right]), n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})(\bar{P}' \mid \bar{Q})\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this last case.

(**Transition triggered by** $\bar{Q}$)

The proof follows similar lines to that of the transitions triggered by $\bar{P}$.

(**Synchronization between** $\bar{P}$ **and** $\bar{Q}$)

We must consider three distinct cases that allow for the synchronization to take place: either the processes are communicating at their upper level ($\uparrow$), or they are communicating at their level ($\downarrow$), or they synchronize on at least one propagated located label.

(*Case upper level*)

We have that

$$(\mathbf{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\tau} (\mathbf{new}\,\widetilde{m},\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.44}$$

where (A.19.44) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} (\mathbf{new}\,\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.45}$$

and (A.19.45) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{v}o)\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.46}$$

and

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{v}o)\overline{\lambda'^{\downarrow}}} n \blacktriangleright \left[\bar{Q}'\right] \tag{A.19.47}$$

We have that (A.19.46) is derived from

$$\bar{P} \xrightarrow{(\widetilde{v}o)\lambda'^{\uparrow}} \bar{P}' \tag{A.19.48}$$

and (A.19.47) is derived from

$$\bar{Q} \xrightarrow{(\widetilde{v}o)\overline{\lambda'^{\uparrow}}} \bar{Q}' \tag{A.19.49}$$

From (A.19.48) and (A.19.49) we conclude

$$\bar{P} \mid \bar{Q} \xrightarrow{\tau} (\mathbf{new}\,\widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.50}$$

From (A.19.50) we conclude

$$(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{\tau} (\mathbf{new}\,\widetilde{m},\widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.51}$$

and finally from (A.19.51) we conclude

$$n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\tau} n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m},\widetilde{o})(\bar{P}' \mid \bar{Q}')\right] \tag{A.19.52}$$

By definition of $\mathcal{R}$ (A.19.1) we conclude

$$((\mathbf{new}\,\widetilde{m},\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]), n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m},\widetilde{o})(\bar{P}' \mid \bar{Q}')\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case process level*)

We have that

$$(\mathbf{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\tau} (\mathbf{new}\,\widetilde{m},\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.53}$$

where (A.19.53) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} (\mathbf{new}\,\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.54}$$

and (A.19.54) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n \blacktriangleright \cdot \lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.55}$$

and

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})n \blacktriangleright \cdot \overline{\lambda'^{\downarrow}}} n \blacktriangleright \left[\bar{Q}'\right] \tag{A.19.56}$$

We have that (A.19.55) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} \bar{P}' \tag{A.19.57}$$

and (A.19.56) is derived from

$$\bar{Q} \xrightarrow{(\widetilde{\nu o})\overline{\lambda'^{\downarrow}}} \bar{Q}' \tag{A.19.58}$$

From (A.19.57) and (A.19.58) we conclude

$$\bar{P} \mid \bar{Q} \xrightarrow{\tau} (\mathbf{new}\ \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.59}$$

From (A.19.59) we conclude

$$(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{\tau} (\mathbf{new}\ \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.60}$$

and finally from (A.19.60) we conclude

$$n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\tau} n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right] \tag{A.19.61}$$

By definition of $\mathcal{R}$ (A.19.1) we conclude

$$((\mathbf{new}\ \widetilde{m}, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]), n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case located label*)

We have that

$$(\mathbf{new}\ \widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\tau} (\mathbf{new}\ \widetilde{m}, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.62}$$

where (A.19.62) is derived from

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} (\mathbf{new}\ \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.63}$$

and (A.19.63) is derived from

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})r\rho\lambda} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.64}$$

and

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\overline{(\widetilde{\nu o})r\rho\lambda}} n \blacktriangleright \left[\bar{Q}'\right] \tag{A.19.65}$$

We have that (A.19.64) is derived from either

$$\bar{P} \xrightarrow{(\widetilde{\nu o})r\rho\lambda} \bar{P}' \tag{A.19.66}$$

or from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda} \bar{P}' \tag{A.19.67}$$

in which case $r = n$ and $\rho = \blacktriangleright$. We also have that (A.19.65) is either derived from

$$\bar{Q} \xrightarrow{\overline{(\widetilde{\nu o})r\rho\lambda}} \bar{Q}' \tag{A.19.68}$$

or from

$$\bar{Q} \xrightarrow{\overline{(\widetilde{\nu o})\lambda}} \bar{Q}' \tag{A.19.69}$$

in which case $r = n$ and $\rho = \blacktriangleright$. We consider the combinatory of the four distinct cases: either (A.19.66) and (A.19.68) are true, or (A.19.66) and (A.19.69) are true, or (A.19.67) and (A.19.68) are true, or finally (A.19.67) and (A.19.69) are true.

In the first case we have (A.19.66) and (A.19.68) from which we conclude

$$\bar{P} \mid \bar{Q} \xrightarrow{\tau} (\mathbf{new}\ \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.70}$$

From (A.19.70) we conclude

$$(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{\tau} (\mathbf{new}\ \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.71}$$

In the second case we have (A.19.67) and (A.19.68) from which we conclude

$$\bar{P} \mid \bar{Q} \xrightarrow{n \blacktriangleright \mathtt{here}} (\mathbf{new}\ \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.72}$$

From (A.19.72) we conclude

$$(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{n \blacktriangleright \mathtt{here}} (\mathbf{new}\ \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.73}$$

In the third case we have (A.19.67) and (A.19.68) from which we conclude

$$\bar{P} \mid \bar{Q} \xrightarrow{n \blacktriangleright \mathtt{here}} (\mathbf{new}\ \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.74}$$

From (A.19.74) we conclude

$$(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{n \blacktriangleright \mathtt{here}} (\mathbf{new}\ \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.75}$$

The fourth case has already been proven given that λ must be either be a $\uparrow$ label or a $\downarrow$ being the proof for these cases shown above (it is not possible that two $\leftarrow$ not located synchronize after being located at the same context, and service instantiation is always a located label).

Finally either from (A.19.71) or (A.19.73) or (A.19.75) we conclude

$$n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\tau} n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right] \tag{A.19.76}$$

By definition of $\mathcal{R}$ (A.19.1) we conclude

$$((\mathbf{new}\ \widetilde{m}, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]), n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this last case.

(**Symmetry**)

We must now consider the same three distinct cases for the derivation of the transition of $n \blacktriangleright \left[(\mathbf{new}\ \widetilde{m})(\bar{P} \mid \bar{Q})\right]$: it is either triggered by $\bar{P}$ or by $\bar{Q}$ or by a synchronization of $\bar{P}$ and $\bar{Q}$. We ommit the proofs for the first two, given that they follow similar lines to what was shown above, and show the proof for the case of the synchronization. We consider the now four different cases that allow for a synchronization to take place: either the processes are trying to communicate

at their level, or at their upper level, or a synchronization takes places on two located labels or finally on a located label and an unlocated label.

(*Case upper level*)

We have that

$$n \blacktriangleright \left[(\mathbf{new}\, \widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\tau} n \blacktriangleright \left[(\mathbf{new}\, \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right] \tag{A.19.77}$$

where (A.19.77) is derived from

$$(\mathbf{new}\, \widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{\tau} (\mathbf{new}\, \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.78}$$

and (A.19.78) is derived from

$$\bar{P} \mid \bar{Q} \xrightarrow{\tau} (\mathbf{new}\, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.79}$$

(A.19.79) is derived from

$$\bar{P} \xrightarrow{(\widetilde{v}o)\lambda'^{\uparrow}} \bar{P}' \tag{A.19.80}$$

and

$$\bar{Q} \xrightarrow{(\widetilde{v}o)\overline{\lambda'^{\uparrow}}} \bar{Q}' \tag{A.19.81}$$

From (A.19.80) we conclude

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{v}o)\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.82}$$

and from (A.19.81) we conclude

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{v}o)\overline{\lambda'^{\downarrow}}} n \blacktriangleright \left[\bar{Q}'\right] \tag{A.19.83}$$

From (A.19.82) and (A.19.83) we conclude

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} (\mathbf{new}\, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.84}$$

and finally from (A.19.84) we conclude

$$(\mathbf{new}\, \widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\tau} (\mathbf{new}\, \widetilde{m}, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.85}$$

By definition of $\mathcal{R}$ (A.19.1) we conclude

$$((\mathbf{new}\, \widetilde{m}, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]), n \blacktriangleright \left[(\mathbf{new}\, \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case process level*)

We have that

$$n \blacktriangleright \left[(\mathbf{new}\, \widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\tau} n \blacktriangleright \left[(\mathbf{new}\, \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right] \tag{A.19.86}$$

where (A.19.86) is derived from

$$(\mathbf{new}\, \widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{\tau} (\mathbf{new}\, \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.87}$$

and (A.19.87) is derived from

$$\bar{P} \mid \bar{Q} \xrightarrow{\tau} (\mathbf{new}\, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.88}$$

(A.19.88) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda'^{\downarrow}} \bar{P}' \tag{A.19.89}$$

and

$$\bar{Q} \xrightarrow{(\widetilde{\nu o})\overline{\lambda'^{\downarrow}}} \bar{Q}' \tag{A.19.90}$$

From (A.19.89) we conclude

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\lambda'^{\downarrow}} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.91}$$

and from (A.19.90) we conclude

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\cdot\overline{\lambda'^{\downarrow}}} n \blacktriangleright \left[\bar{Q}'\right] \tag{A.19.92}$$

From (A.19.91) and (A.19.92) we conclude

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} (\mathbf{new}\, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.93}$$

and finally from (A.19.93) we conclude

$$(\mathbf{new}\, \widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\tau} (\mathbf{new}\, \widetilde{m}, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.94}$$

By definition of $\mathcal{R}$ (A.19.1) we conclude

$$((\mathbf{new}\, \widetilde{m}, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]), n \blacktriangleright \left[(\mathbf{new}\, \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case synchronization on complete labels*)

We have that

$$n \blacktriangleright \left[(\mathbf{new}\, \widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\tau} n \blacktriangleright \left[(\mathbf{new}\, \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}')\right] \tag{A.19.95}$$

where (A.19.95) is derived from

$$(\mathbf{new}\, \widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{\tau} (\mathbf{new}\, \widetilde{m}, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.96}$$

and (A.19.96) is derived from

$$\bar{P} \mid \bar{Q} \xrightarrow{\tau} (\mathbf{new}\, \widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.97}$$

(A.19.97) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})r\rho\lambda} \bar{P}' \tag{A.19.98}$$

and

$$\bar{Q} \xrightarrow{\overline{(\widetilde{\nu o})r\rho\lambda}} \bar{Q}' \tag{A.19.99}$$

From (A.19.98) we conclude

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})r\rho\lambda} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.100}$$

and from (A.19.99) we conclude

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\overline{(\widetilde{\nu o})r\rho\lambda}} n \blacktriangleright \left[\bar{Q}'\right] \tag{A.19.101}$$

From (A.19.100) and (A.19.101) we conclude

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} (\mathbf{new}\, \widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.102}$$

and finally from (A.19.102) we conclude

$$(\mathbf{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\tau} (\mathbf{new}\,\widetilde{m},\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.103}$$

By definition of $\mathcal{R}$ (A.19.1) we conclude

$$((\mathbf{new}\,\widetilde{m},\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]), n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m},\widetilde{o})(\bar{P}' \mid \bar{Q}')\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this case.

(*Case synchronization on incomplete labels*)

We have that

$$n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q})\right] \xrightarrow{\tau} n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m},\widetilde{o})(\bar{P}' \mid \bar{Q}')\right] \tag{A.19.104}$$

where (A.19.104) is derived from

$$(\mathbf{new}\,\widetilde{m})(\bar{P} \mid \bar{Q}) \xrightarrow{n\blacktriangleright\mathtt{here}} (\mathbf{new}\,\widetilde{m},\widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.105}$$

and (A.19.105) is derived from

$$\bar{P} \mid \bar{Q} \xrightarrow{n\blacktriangleright\mathtt{here}} (\mathbf{new}\,\widetilde{o})(\bar{P}' \mid \bar{Q}') \tag{A.19.106}$$

(A.19.106) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu o})\lambda} \bar{P}' \tag{A.19.107}$$

and

$$\bar{Q} \xrightarrow{\overline{(\widetilde{\nu o})n\blacktriangleright\lambda}} \bar{Q}' \tag{A.19.108}$$

From (A.19.107) we conclude

$$n \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu o})n\blacktriangleright\lambda} n \blacktriangleright \left[\bar{P}'\right] \tag{A.19.109}$$

and from (A.19.108) we conclude

$$n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\overline{(\widetilde{\nu o})n\blacktriangleright\lambda}} n \blacktriangleright \left[\bar{Q}'\right] \tag{A.19.110}$$

From (A.19.109) and (A.19.110) we conclude

$$n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right] \xrightarrow{\tau} (\mathbf{new}\,\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.111}$$

and finally from (A.19.111) we conclude

$$(\mathbf{new}\,\widetilde{m})(n \blacktriangleright [\bar{P}] \mid n \blacktriangleright \left[\bar{Q}\right]) \xrightarrow{\tau} (\mathbf{new}\,\widetilde{m},\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]) \tag{A.19.112}$$

By definition of $\mathcal{R}$ (A.19.1) we conclude

$$((\mathbf{new}\,\widetilde{m},\widetilde{o})(n \blacktriangleright \left[\bar{P}'\right] \mid n \blacktriangleright \left[\bar{Q}'\right]), n \blacktriangleright \left[(\mathbf{new}\,\widetilde{m},\widetilde{o})(\bar{P}' \mid \bar{Q}')\right]) \in \mathcal{R} \cup \sim$$

which completes the proof for this last case.

Lemma A.20 *We have $n \blacktriangleleft [P] \mid n \blacktriangleleft [Q] \sim n \blacktriangleleft [P \mid Q]$, for any name n and processes P, Q.*

Proof. Analogous to that of Lemma A.19.

Lemma A.21 *We have that $m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [P]]] \sim n \blacktriangleright [o \blacktriangleright [P]]$, for any names m, n, o.*

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [P]]], n \blacktriangleright [o \blacktriangleright [P]]) \mid \forall m,n,o\} \cup \{(P,P)\} \tag{A.21.1}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R}$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R}$.

We must consider two different cases: either $(P,Q) \in \{(P,P)\}$ or not.

If $(P,Q) \in \{(P,P)\}$ we we directly have that for any transition $P \xrightarrow{\lambda} P'$ it is the case that $(P',P') \in \mathcal{R}$.

Otherwise we have that P and Q are of the form $m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]]$ and $n \blacktriangleright [o \blacktriangleright [\bar{P}]]$, respectively, for some $m,n,o,\bar{P}$.

We must consider seven different derivations for the transitions of $m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]]$, so we have that λ is either derived from `throw`, or from $(\widetilde{\nu p})q\rho\lambda'$ such that $\lambda' \neq$ `here`, or from $o \blacktriangleright$ `here`, or from $(\widetilde{\nu p})\lambda'^{\leftarrow}$, or from $(\widetilde{\nu p})\lambda'^{\downarrow}$, or from $(\widetilde{\nu p})\lambda'^{\uparrow}$ or finally from (νc)`def` s.

(*Case* `throw`)

We have that

$$m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]] \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.21.2}$$

where (A.21.2) is derived from

$$n \blacktriangleright [o \blacktriangleright [\bar{P}]] \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.21.3}$$

and (A.21.3) is derived from

$$o \blacktriangleright [\bar{P}] \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.21.4}$$

and (A.21.4) is derived from

$$\bar{P} \xrightarrow{\texttt{throw}} \bar{P}' \tag{A.21.5}$$

By definition of $\mathcal{R}$ (A.21.1) we have that

$$(\bar{P}', \bar{P}') \in \mathcal{R}$$

which along with (A.21.3) completes the proof for this case.

(*Case* $((\widetilde{\nu p})q\rho\lambda'$)

We have that

$$m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]] \xrightarrow{((\widetilde{\nu p})q\rho\lambda'} m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right] \tag{A.21.6}$$

where (A.21.6) is derived from

$$n \blacktriangleright [o \blacktriangleright [\bar{P}]] \xrightarrow{((\widetilde{\nu p})q\rho\lambda'} n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right] \tag{A.21.7}$$

and (A.21.7) is derived from

$$o \blacktriangleright [\bar{P}] \xrightarrow{((\widetilde{\nu p})q\rho\lambda'} o \blacktriangleright \left[\bar{P}'\right] \tag{A.21.8}$$

and (A.21.8) is derived from

$$\bar{P} \xrightarrow{((\widetilde{\nu p})q\rho\lambda'} \bar{P}' \tag{A.21.9}$$

By definition of $\mathcal{R}$ (A.21.1) we have that

$$(m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right], n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]) \in \mathcal{R}$$

which along with (A.21.7) completes the proof for this case.

(*Case* $o \blacktriangleright \texttt{here}$)

We have that

$$m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]] \xrightarrow{\tau} m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right] \tag{A.21.10}$$

where (A.21.10) is derived from

$$n \blacktriangleright [o \blacktriangleright [\bar{P}]] \xrightarrow{\tau} n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right] \tag{A.21.11}$$

and (A.21.11) is derived from

$$o \blacktriangleright [\bar{P}] \xrightarrow{\tau} o \blacktriangleright \left[\bar{P}'\right] \tag{A.21.12}$$

and (A.21.12) is derived from

$$\bar{P} \xrightarrow{o \blacktriangleright \texttt{here}} \bar{P}' \tag{A.21.13}$$

By definition of $\mathcal{R}$ (A.21.1) we have that

$$(m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right], n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]) \in \mathcal{R}$$

which along with (A.21.11) completes the proof for this case.

(*Case* $(\widetilde{\nu p})\lambda'^{\leftarrow}$)

We have that

$$m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]] \xrightarrow{(\widetilde{\nu p})o \blacktriangleright \cdot \lambda'^{\leftarrow}} m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right] \tag{A.21.14}$$

where (A.21.14) is derived from

$$n \blacktriangleright [o \blacktriangleright [\bar{P}]] \xrightarrow{(\widetilde{\nu p})o \blacktriangleright \cdot \lambda'^{\leftarrow}} n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right] \tag{A.21.15}$$

and (A.21.15) is derived from

$$o \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu p})o \blacktriangleright \cdot \lambda'^{\leftarrow}} o \blacktriangleright \left[\bar{P}'\right] \tag{A.21.16}$$

and (A.21.16) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu p})\lambda'^{\leftarrow}} \bar{P}' \tag{A.21.17}$$

By definition of $\mathcal{R}$ (A.21.1) we have that

$$(m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right], n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]) \in \mathcal{R}$$

which along with (A.21.15) completes the proof for this case.

(*Case* $(\widetilde{\nu p})\lambda'^{\downarrow}$)

We have that

$$m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]] \xrightarrow{(\widetilde{\nu p})o \blacktriangleright \cdot \lambda'^{\downarrow}} m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right] \tag{A.21.18}$$

where (A.21.18) is derived from

$$n \blacktriangleright [o \blacktriangleright [\bar{P}]] \xrightarrow{(\widetilde{\nu p})o \blacktriangleright \cdot \lambda'^{\downarrow}} n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right] \tag{A.21.19}$$

and (A.21.19) is derived from

$$o \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu p})o\blacktriangleright\cdot\lambda'^{\downarrow}} o \blacktriangleright \left[\bar{P}'\right] \tag{A.21.20}$$

and (A.21.20) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu p})\lambda'^{\downarrow}} \bar{P}' \tag{A.21.21}$$

By definition of $\mathcal{R}$ (A.21.1) we have that

$$\left(m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right], n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right) \in \mathcal{R}$$

which along with (A.21.19) completes the proof for this case.

(*Case* $(\widetilde{\nu p})\lambda'^{\uparrow}$)

We have that

$$m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]] \xrightarrow{(\widetilde{\nu p})n\blacktriangleright\cdot\lambda'^{\downarrow}} m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right] \tag{A.21.22}$$

where (A.21.22) is derived from

$$n \blacktriangleright [o \blacktriangleright [\bar{P}]] \xrightarrow{(\widetilde{\nu p})n\blacktriangleright\cdot\lambda'^{\downarrow}} n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right] \tag{A.21.23}$$

and (A.21.23) is derived from

$$o \blacktriangleright [\bar{P}] \xrightarrow{(\widetilde{\nu p})\lambda'^{\downarrow}} o \blacktriangleright \left[\bar{P}'\right] \tag{A.21.24}$$

and (A.21.24) is derived from

$$\bar{P} \xrightarrow{(\widetilde{\nu p})\lambda'^{\uparrow}} \bar{P}' \tag{A.21.25}$$

By definition of $\mathcal{R}$ (A.21.1) we have that

$$\left(m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right], n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right) \in \mathcal{R}$$

which along with (A.21.23) completes the proof for this case.

(*Case* $(\nu c)\texttt{def}\ s$)

We have that

$$m \blacktriangleright [n \blacktriangleright [o \blacktriangleright [\bar{P}]]] \xrightarrow{(\nu c)o\blacktriangleright\texttt{def}\ s} m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right] \tag{A.21.26}$$

where (A.21.26) is derived from

$$n \blacktriangleright [o \blacktriangleright [\bar{P}]] \xrightarrow{(\nu c)o\blacktriangleright\texttt{def}\ s} n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right] \tag{A.21.27}$$

and (A.21.27) is derived from

$$o \blacktriangleright [\bar{P}] \xrightarrow{(\nu c)o\blacktriangleright\texttt{def}\ s} o \blacktriangleright \left[\bar{P}'\right] \tag{A.21.28}$$

and (A.21.28) is derived from

$$\bar{P} \xrightarrow{(\nu c)\texttt{def}\ s} \bar{P}' \tag{A.21.29}$$

By definition of $\mathcal{R}$ (A.21.1) we have that

$$\left(m \blacktriangleright \left[n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right], n \blacktriangleright \left[o \blacktriangleright \left[\bar{P}'\right]\right]\right) \in \mathcal{R}$$

which along with (A.21.27) completes the proof for this last case.

Lemma A.22 *We have that* $m\rho_1[n\rho_2[o\rho_3[P]]] \sim n\rho_2[o\rho_3[P]]$ *, for any names* m,n,o *and polarities* ρ_1,ρ_2,ρ_3.

Proof. Analogous to the proof of Lemma A.21

Lemma A.23 *We have* $n\rho[\mathtt{stop}] \sim \mathtt{stop}$.

Proof. The result is a direct outcome of the fact that neither of the systems has transitions.

Lemma A.24 *We have* $n \blacktriangleright [\mathtt{out} \uparrow l(\widetilde{v}).R] \sim \mathtt{out} \downarrow l(\widetilde{v}).n \blacktriangleright [R]$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(n \blacktriangleright [\mathtt{out} \uparrow l(\widetilde{v}).R], \mathtt{out} \downarrow l(\widetilde{v}).n \blacktriangleright [R]) \mid \forall n,l,\widetilde{v},R\} \cup \{(P,P)\} \tag{A.24.1}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R}$. We must show that for every P',λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R}$.

We must consider two different cases: either $(P,Q) \in \{(n \blacktriangleright [\mathtt{out} \uparrow l(\widetilde{v}).R], \mathtt{out} \downarrow l(\widetilde{v}).n \blacktriangleright [R])\}$ or $(P,Q) \in \{(P,P)\}$.

If $(P,Q) \in \{(P,P)\}$ we we directly have that for any transition $P \xrightarrow{\lambda} P'$ it is the case that $(P',P') \in \mathcal{R}$.

If $(P,Q) \in \{(n \blacktriangleright [\mathtt{out} \uparrow l(\widetilde{v}).R], \mathtt{out} \downarrow l(\widetilde{v}).n \blacktriangleright [R])\}$ we have that P is of the form $n \blacktriangleright [\mathtt{out} \uparrow l(\widetilde{v}).R]$ and Q is of the form $\mathtt{out} \downarrow l(\widetilde{v}).n \blacktriangleright [R]$, for some process R and names $n,\widetilde{v}$ and label l.

We must consider the only possible transition of $n \blacktriangleright [\mathtt{out} \uparrow l(\widetilde{v}).R]$:

$$n \blacktriangleright [\mathtt{out} \uparrow l(\widetilde{v}).R] \xrightarrow{\overline{\downarrow l(\widetilde{v})}} n \blacktriangleright [R] \tag{A.24.2}$$

where (A.24.2) is derived from

$$\mathtt{out} \uparrow l(\widetilde{v}).R \xrightarrow{\overline{\uparrow l(\widetilde{v})}} R \tag{A.24.3}$$

We can derive a matching transition to (A.24.2) of $\mathtt{out} \downarrow l(\widetilde{v}).n \blacktriangleright [R]$ (which is also the only possible one it can perform)

$$\mathtt{out} \downarrow l(\widetilde{v}).n \blacktriangleright [R] \xrightarrow{\overline{\downarrow l(\widetilde{v})}} n \blacktriangleright [R] \tag{A.24.4}$$

By definition of $\mathcal{R}$ (A.24.1) we directly have that

$$(n \blacktriangleright [R], n \blacktriangleright [R]) \in \mathcal{R}$$

which completes the proof.

Lemma A.25 *We have* $n \blacktriangleleft [\mathtt{out} \uparrow m(\widetilde{v}).R] \sim \mathtt{out} \downarrow m(\widetilde{v}).n \blacktriangleleft [R]$.

Proof. Analogous to that of Lemma A.24.

Lemma A.26 *We have* $n \blacktriangleright [\mathtt{in} \uparrow m(\widetilde{x}).R] \sim \mathtt{in} \downarrow m(\widetilde{x}).n \blacktriangleright [R]$ *(*$n \notin \widetilde{x}$*).*

Proof. Analogous to that of Lemma A.24.

Lemma A.27 *We have* $n \blacktriangleleft [\mathtt{in} \uparrow m(\widetilde{x}).R] \sim \mathtt{in} \downarrow m(\widetilde{x}).n \blacktriangleleft [R]$ *(*$n \notin \widetilde{x}$*).*

Proof. Analogous to that of Lemma A.24.

Lemma A.28 *We have* $m \blacktriangleright [n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).R]] \sim n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]]$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(m \blacktriangleright [n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).R]], n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]]) \mid \forall n, m, \widetilde{v}, l, R\} \cup \sim \quad \text{(A.28.1)}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R}$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R}$.

We must consider two different cases: either $(P,Q) \in \{(m \blacktriangleright [n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).R]], n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]])\}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \{(m \blacktriangleright [n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).R]], n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]])\}$ we have that P is of the form $m \blacktriangleright [n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).R]]$ and Q is of the form $n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]]$, for some process R and names $n, m, \widetilde{v}$ and label l.

We must consider the only possible transition of $m \blacktriangleright [n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).R]]$:

$$m \blacktriangleright [n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).R]] \xrightarrow{\overline{n \blacktriangleright \downarrow l(\widetilde{v})}} m \blacktriangleright [n \blacktriangleright [R]] \quad \text{(A.28.2)}$$

We can derive a matching transition to (A.28.2) of $n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]]$ (which is also the only possible one it can perform)

$$n \blacktriangleright [\mathtt{out} \downarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]] \xrightarrow{\overline{n \blacktriangleright \downarrow l(\widetilde{v})}} n \blacktriangleright [m \blacktriangleright [n \blacktriangleright [R]]] \quad \text{(A.28.3)}$$

By definition of $\mathcal{R}$ (A.28.1) and Lemma A.22 we have that

$$(m \blacktriangleright [n \blacktriangleright [R]], n \blacktriangleright [m \blacktriangleright [n \blacktriangleright [R]]]) \in \mathcal{R}$$

which completes the proof.

Lemma A.29 *We have* $m\, \rho_1\, [n\, \rho_2\, [\mathtt{out} \downarrow l(\widetilde{v}).R]] \sim n\, \rho_2\, [\mathtt{out} \downarrow l(\widetilde{v}).m\, \rho_1\, [n\, \rho_2\, [R]]]$.

Proof. Analogous to that of Lemma A.28.

Lemma A.30 *We have* $m \blacktriangleright [n \blacktriangleright [\mathtt{in} \downarrow l(\widetilde{x}).P]] \sim n \blacktriangleright [\mathtt{in} \downarrow l(\widetilde{x}).m \blacktriangleright [n \blacktriangleright [P]]]$ *(*$m, n \notin \widetilde{x}$*).*

Proof. Analogous to that of Lemma A.28.

Lemma A.31 *We have* $m\ \rho_1\,[n\ \rho_2\,[\mathtt{in} \downarrow l(\widetilde{x}).P]] \sim n\ \rho_2\,[\mathtt{in} \downarrow l(\widetilde{x}).m\ \rho_1\,[n\ \rho_2\,[P]]]$ *(*$m,n \notin \widetilde{x}$*).*

Proof. Analogous to that of Lemma A.28.

Lemma A.32 *We have* $m \blacktriangleright [n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).R]] \sim n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]]$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(m \blacktriangleright [n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).R]], n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]]) \mid \forall n,m,\widetilde{v},l,R\} \cup \sim \quad \text{(A.32.1)}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R}$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R}$.

We must consider two different cases: either $(P,Q) \in \{(m \blacktriangleright [n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).R]], n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]])\}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \{(m \blacktriangleright [n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).R]], n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]])\}$ we have that P is of the form $m \blacktriangleright [n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).R]]$ and Q is of the form $n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]]$, for some process R and names $n,m,\widetilde{v}$ and label l.

We must consider the only possible transition of $m \blacktriangleright [n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).R]]$:

$$m \blacktriangleright [n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).R]] \xrightarrow{\overline{n \blacktriangleright \leftarrow l(\widetilde{v})}} m \blacktriangleright [n \blacktriangleright [R]] \quad \text{(A.32.2)}$$

We can derive a matching transition to (A.32.2) of $n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]]$ (which is also the only possible one it can perform)

$$n \blacktriangleright [\mathtt{out} \leftarrow l(\widetilde{v}).m \blacktriangleright [n \blacktriangleright [R]]] \xrightarrow{\overline{n \blacktriangleright \leftarrow l(\widetilde{v})}} n \blacktriangleright [m \blacktriangleright [n \blacktriangleright [R]]] \quad \text{(A.32.3)}$$

By definition of $\mathcal{R}$ (A.32.1) and Lemma A.22 we have that

$$(m \blacktriangleright [n \blacktriangleright [R]], n \blacktriangleright [m \blacktriangleright [n \blacktriangleright [R]]]) \in \mathcal{R}$$

which completes the proof.

Lemma A.33 *We have* $m\ \rho_1\,[n\ \rho_2\,[\mathtt{out} \leftarrow l(\widetilde{v}).R]] \sim n\ \rho_2\,[\mathtt{out} \leftarrow l(\widetilde{v}).m\ \rho_1\,[n\ \rho_2\,[R]]]$.

Proof. Analogous to that of Lemma A.32.

Lemma A.34 *We have* $m \blacktriangleright [n \blacktriangleright [\mathtt{in} \leftarrow l(\widetilde{x}).P]] \sim n \blacktriangleright [\mathtt{in} \leftarrow l(\widetilde{x}).m \blacktriangleright [n \blacktriangleright [P]]]$ *(*$m,n \notin \widetilde{x}$*).*

Proof. Analogous to that of Lemma A.32.

Lemma A.35 *We have* $m\ \rho_1\,[n\ \rho_2\,[\mathtt{in} \leftarrow l(\widetilde{x}).P]] \sim n\ \rho_2\,[\mathtt{in} \leftarrow l(\widetilde{x}).m\ \rho_1\,[n\ \rho_2\,[P]]]$ *(*$m, n \notin \widetilde{x}$*).*

Proof. Analogous to that of Lemma A.32.

Lemma A.36 *We have* $m \blacktriangleright [n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]] \sim n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(m \blacktriangleright [n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]], n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]) \mid \forall n, m, s, R\} \cup \sim \tag{A.36.1}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R}$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \mathcal{R}$.

We must consider two different cases: either $(P,Q) \in \{(m \blacktriangleright [n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]], n \blacktriangleright [\mathtt{def}\ s \Rightarrow R])\}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P', Q') \in \sim$ and hence $(P', Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \{(m \blacktriangleright [n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]], n \blacktriangleright [\mathtt{def}\ s \Rightarrow R])\}$ we have that P is of the form $m \blacktriangleright [n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]]$ and Q is of the form $n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]$, for some process R and names n, m and label s.

We must consider the only possible transition of $m \blacktriangleright [n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]]$:

$$m \blacktriangleright [n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]] \xrightarrow{(\nu c) n \blacktriangleright \mathtt{def}\ s} m \blacktriangleright [n \blacktriangleright [c \blacktriangleright [R]]] \tag{A.36.2}$$

We can derive a matching transition to (A.36.2) of $n \blacktriangleright [\mathtt{def}\ s \Rightarrow R]$ (which is also the only possible one it can perform)

$$n \blacktriangleright [\mathtt{def}\ s \Rightarrow R] \xrightarrow{(\nu c) n \blacktriangleright \mathtt{def}\ s} n \blacktriangleright [c \blacktriangleright [R]] \tag{A.36.3}$$

By definition of $\mathcal{R}$ (A.36.1) and Lemma A.22 we have that

$$(m \blacktriangleright [n \blacktriangleright [c \blacktriangleright [R]]], n \blacktriangleright [c \blacktriangleright [R]]) \in \mathcal{R}$$

which completes the proof.

Lemma A.37 *We have* $m\ \rho_1\,[n\ \rho_2\,[\mathtt{def}\ s \Rightarrow R]] \sim n\ \rho_2\,[\mathtt{def}\ s \Rightarrow R]$.

Proof. Analogous to that of Lemma A.36.

Lemma A.38 *We have* $m \blacktriangleright [n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]] \sim n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]$.

Proof. Let us consider $\mathcal{R}$ defined as

$$\mathcal{R} \triangleq \{(m \blacktriangleright [n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]], n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]) \mid \forall n,m,o,s,R\} \cup \sim \tag{A.38.1}$$

We show that $\mathcal{R}$ is a bisimulation by coinduction on the definition of bisimulation.

Let us consider $(P,Q) \in \mathcal{R}$. We must show that for every P', λ such that

$$P \xrightarrow{\lambda} P'$$

then there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \mathcal{R}$.

We must consider two different cases: either $(P,Q) \in \{(m \blacktriangleright [n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]], n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R])\}$ or $(P,Q) \in \sim$.

If $(P,Q) \in \sim$ we directly have that there exists Q' such that

$$Q \xrightarrow{\lambda} Q'$$

and $(P',Q') \in \sim$ and hence $(P',Q') \in \mathcal{R} \cup \sim$.

If $(P,Q) \in \{(m \blacktriangleright [n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]], n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R])\}$ we have that P is of the form $m \blacktriangleright [n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]]$ and Q is of the form $n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]$, for some process R and names n,m,o, role ρ and label s.

We must consider the only possible transition of $m \blacktriangleright [n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]]$:

$$m \blacktriangleright [n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]] \xrightarrow{\overline{(\nu c) o\ \rho\ \mathtt{def}\ s}} m \blacktriangleright [n \blacktriangleright [c \blacktriangleleft [R]]] \tag{A.38.2}$$

We can derive a matching transition to (A.38.2) of $n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R]$ (which is also the only possible one it can perform)

$$n \blacktriangleright [\mathtt{instance}\ o\ \rho\ s \Leftarrow R] \xrightarrow{\overline{(\nu c) o\ \rho\ \mathtt{def}\ s}} n \blacktriangleright [c \blacktriangleleft [R]] \tag{A.38.3}$$

By definition of $\mathcal{R}$ (A.38.1) and Lemma A.22 we have that

$$(m \blacktriangleright [n \blacktriangleright [c \blacktriangleleft [R]]], n \blacktriangleright [c \blacktriangleleft [R]]) \in \mathcal{R}$$

which completes the proof.

Lemma A.39 *We have* $m\ \rho_1\ [n\ \rho_2\ [\mathtt{instance}\ o\ \rho_3\ s \Leftarrow R]] \sim n\ \rho_2\ [\mathtt{instance}\ o\ \rho_3\ s \Leftarrow R]$.

Proof. Analogous to that of Lemma A.38.

Proposition 5.5 proof

Proof. We proceed by induction on the number of guarded processes. In the proof we implicitly consider Theorem 5.2. Consider that P is of the form

$$n\rho_1\,[\ldots m\rho_2\,[o\rho_3\,[G \mid P_1] \mid P_2] \ldots \mid P_3] \mid P' \tag{A.39.1}$$

for some guarded process G, processes $P_1, P_2, \ldots, P_3, P'$, names $n, \ldots, m, o$, and roles $\rho_1, \ldots, \rho_2, \rho_3$. Considering Proposition 5.4(2) and (A.39.1) we have that

$$\begin{array}{c} n\rho_1\,[\ldots m\rho_2\,[o\rho_3\,[G \mid P_1] \mid P_2] \ldots \mid P_3] \mid P' \\ \sim \\ n\rho_1\,[\ldots m\rho_2\,[o\rho_3\,[G] \mid o\rho_3\,[P_1] \mid P_2] \ldots \mid P_3] \mid P' \end{array} \tag{A.39.2}$$

We apply the split (5.4(2)) as in (A.39.2) repeatedly up to top level and obtain

$$\begin{array}{c} n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[G \mid P_1\right] \mid P_2\right]\ldots \mid P_3\right] \mid P' \\ \sim \\ n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[G\right]\right]\ldots\right] \\ \mid n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[P_1\right] \mid P_2\right]\ldots \mid P_3\right] \mid P' \end{array} \tag{A.39.3}$$

After which we consider Proposition 5.4(3) repeatedly and obtain

$$\begin{array}{c} n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[G \mid P_1\right] \mid P_2\right]\ldots \mid P_3\right] \mid P' \\ \sim \\ m\rho_2\left[o\rho_3\left[G\right]\right] \\ \mid n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[P_1\right] \mid P_2\right]\ldots \mid P_3\right] \mid P' \end{array} \tag{A.39.4}$$

By i.h. on

$$n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[P_1\right] \mid P_2\right]\ldots \mid P_3\right] \mid P' \tag{A.39.5}$$

we obtain there exists a process $\bar{P}$ such that $\bar{P}$ is in normal form and

$$\bar{P} \sim n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[P_1\right] \mid P_2\right]\ldots \mid P_3\right] \mid P' \tag{A.39.6}$$

From (A.39.4) and (A.39.6) we conclude

$$\begin{array}{c} n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[G \mid P_1\right] \mid P_2\right]\ldots \mid P_3\right] \mid P' \\ \sim \\ m\rho_2\left[o\rho_3\left[G\right]\right] \mid \bar{P} \end{array} \tag{A.39.7}$$

We now must ensure uniqueness of the sequence (m, ρ_2, o, ρ_3). We have that if there exists such a sequence in $\bar{P}$ then it is unique since $\bar{P}$ is in normal form. Let us consider there exists $\bar{P}'$ and G' such that $\bar{P}$ is of the form

$$\bar{P}' \mid m\rho_2\left[o\rho_3\left[G'\right]\right] \tag{A.39.8}$$

From (A.39.7) and (A.39.8) we conclude

$$\begin{array}{c} m\rho_2\left[o\rho_3\left[G\right]\right] \mid \bar{P} \sim \\ m\rho_2\left[o\rho_3\left[G\right]\right] \mid \bar{P}' \mid m\rho_2\left[o\rho_3\left[G'\right]\right] \end{array} \tag{A.39.9}$$

From (A.39.9) and Proposition 5.4(2) we conclude

$$\begin{array}{c} m\rho_2\left[o\rho_3\left[G\right]\right] \mid \bar{P}' \mid m\rho_2\left[o\rho_3\left[G'\right]\right] \sim \\ m\rho_2\left[o\rho_3\left[G\right] \mid o\rho_3\left[G'\right]\right] \mid \bar{P}' \end{array} \tag{A.39.10}$$

Again considering Proposition 5.4(2), from (A.39.10) we conclude

$$\begin{array}{c} m\rho_2\left[o\rho_3\left[G\right] \mid o\rho_3\left[G'\right]\right] \mid \bar{P}' \sim \\ m\rho_2\left[o\rho_3\left[G \mid G'\right]\right] \mid \bar{P}' \end{array} \tag{A.39.11}$$

The uniqueness of the (n, ρ) sequences is proved in the same way. From (A.39.11) and (A.39.10) and (A.39.9) and (A.39.8) and (A.39.7) we conclude

$$\begin{array}{c} n\rho_1\left[\ldots m\rho_2\left[o\rho_3\left[G \mid P_1\right] \mid P_2\right]\ldots \mid P_3\right] \mid P' \\ \sim \\ m\rho_2\left[o\rho_3\left[G \mid G'\right]\right] \mid \bar{P}' \end{array} \tag{A.39.12}$$

which completes the proof.

Proof of auxiliar results to Proposition 4.1

We consider the semantics of Orc as defined in the following table (from [16]).

$$\frac{u \text{ fresh}}{p.S(c) \xrightarrow{p.S<c,u>} ?u} \qquad ?u \xrightarrow{u?c} let(c) \qquad let(c) \xrightarrow{!c} 0$$

$$\frac{f \xrightarrow{l} f' \quad l \neq !c}{f \gg x \gg g \xrightarrow{l} f' \gg x \gg g} \qquad \frac{f \xrightarrow{!c} f'}{f \gg x \gg g \xrightarrow{\tau} (f' \gg x \gg g) \mid (g\{x \leftarrow c\})}$$

$$\frac{f \xrightarrow{l} f'}{f \mid g \xrightarrow{l} f' \mid g} \qquad \frac{f \xrightarrow{l} f' \quad l \neq !c}{g \ \mathtt{where}\ x :\in f \xrightarrow{l} g \ \mathtt{where}\ x :\in f'}$$

$$\frac{g \xrightarrow{l} g'}{f \mid g \xrightarrow{l} f \mid g'} \qquad \frac{g \xrightarrow{l} g'}{g \ \mathtt{where}\ x :\in f \xrightarrow{l} g' \ \mathtt{where}\ x :\in f}$$

$$\frac{\{n.S(x) = e\} \in D}{n.S(c) \xrightarrow{\tau} e\{x \leftarrow c\}} \qquad \frac{f \xrightarrow{!c} f'}{g \ \mathtt{where}\ x :\in f \xrightarrow{\tau} g\{x \leftarrow c\}}$$

We prove some auxiliar results.

Lemma A.40 *Let O be an orc process. We have that for any sequence of transitions $\widetilde{\lambda}$ such that*

$$[\![O]\!]_{out} \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{k-1}} P' \xrightarrow{\lambda_k} \ldots$$

Then for any $i \in 1, \ldots, k, \ldots$ we have that either $\lambda_i = \tau$ or λ is a located transition label and $out \notin fn(\lambda)$ or $\lambda_i = \overline{\downarrow out(c)}$, for some c.

Proof. (Sketch) In $[\![O]\!]_{out}$ all contexts are either anonymous or their name is restricted. Restricted and anonymous contexts do not hold any observable behavior on $\downarrow$ or $\leftarrow$ labels. This leaves out $\uparrow$ transition labels that become $\downarrow$ when crossing a context barrier, which we can verify to always occur along with the *out* parameter in $[\![O]\!]_{out}$: hence $[\![O]\!]_{out}$ may hold $\overline{\downarrow out(c)}$ transitions, for some name c.

We must also consider *def* transitions of $[\![O]\!]_{out}$. We have that the all the possible behaviors that originate from a site call are $\overline{(\nu u)n \blacktriangleright def\ S}$, for some n and S, $\overline{u \blacktriangleleft\!\leftarrow args(c)}$ for some u, c, and $u \blacktriangleleft\!\leftarrow \overline{result(c)}$ for some c (which are all located transition labels where *out* does not occur) and finally $\overline{\downarrow out(c)}$ for some c.

Lemma A.41 *Let O be an orc process. We have that*

$$[\![O]\!]_a \approx [\![O]\!]_b\{b \leftarrow a\}$$

Proof. Direct from the definition of the encoding.

Lemma A.42 *Let O be an Orc process. We have that*

$$[\![O]\!]_{out} \approx_{out} [[\![O]\!]_{out_1} \mid !\ \mathtt{in} \downarrow out_1(x).\mathtt{out} \uparrow out(x)]$$

Proof. (Sketch) Let us consider every possible transition of $[\![O]\!]_{out}$.

By Lemma A.40 we have that $[\![O]\!]_{out}$ holds transitions that hold either τ as transition label, or a located transition label λ where the parameter *out* does not ocurr, or $\overline{\downarrow out(c)}$ for some name c.

For the case of a τ transition we can also derive a τ transition for $[\![O]\!]_{out_1}$, considering Lemma A.41. For the case of located transition label, where the encoding parameter does not occur, we can also derive a matching transition of $[\![O]\!]_{out_1}$.

For the case of a $\overline{\downarrow out(c)}$ we have that $[\![O]\!]_{out_1}$ has a $\overline{\downarrow out_1(c)}$ transition and hence after one τ step $[[\![O]\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).\mathbf{out} \uparrow out(x)]$ may exhibit a $\overline{\downarrow out(c)}$ transition ($\overset{\overline{\downarrow out(c)}}{\Rightarrow}$). The proof of the symmetric cases is analogous.

Lemma A.43 *Let f, g be a Orc processes. We have that*

$$\begin{array}{l}[[\![f]\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)) \\ \mid [\![g\{x{\leftarrow}c\}]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)] \\ \approx_{out} \\ [[\![f]\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \\ \mid [\![g\{x{\leftarrow}c\}]\!]_{out}\end{array}$$

Proof. Follows the lines of the proof of Lemma A.42.

We denote by $\mathcal{C}[\cdot]$ a conversation calculus process term with the occurrence of a hole $\cdot$, and $\mathcal{C}[\mathcal{P}]$ the term obtained by replacing the hole with process P.

Lemma A.44 *Let f be an Orc process. We have that*

$$\begin{array}{l}(\mathbf{new}\, a)(\mathcal{C}[a \blacktriangleleft [!\,\mathbf{in} \leftarrow result(x).\mathbf{out} \uparrow out(x)]] \\ \qquad\qquad \mid a \blacktriangleright [[\![f]\!]_{out} \mid !\,\mathbf{in} \downarrow out(x).\mathbf{out} \leftarrow result(x)]) \\ \approx \\ \mathcal{C}[[\![f]\!]_{out}]\end{array}$$

Proof. (Sketch) Follows the lines of Lemma A.42. Process $[\![f]\!]_{out}$ can either interact with the external environment (by means of located transition labels) or internally evolve or output in *out*. Interactions with the external environment and internal evolutions can take place regardless of the placement of the process $[\![f]\!]_{out}$ in the term. Outputs in *out* are forwarded to the endpoint that outputs in *out* in the environment defined by its placement in the term, hence acting the same as the process $[\![f]\!]_{out}$ with the same placement in the term.

Lemma A.45 *Let P be a process such that the only possible observations over P hold transition labels located at $n\rho$, for a specific name n and some role ρ. Then we have that*

$$(\mathbf{new}\, n)P \approx \mathbf{stop}$$

Proof. The result is immediate. $(\mathbf{new}\, n)P$ has no observable behavior.

Lemma A.46 *We have*

$$[\![O]\!]_{out}\{x{\leftarrow}c\} \approx [\![O\{x{\leftarrow}c\}]\!]_{out}$$

Proof. By induction on the structure of O. We prove the interesting case of the **where** construct. We have that

$$\begin{array}{rcl}[\![f\ \mathbf{where}\ x :\in g]\!] & \triangleq & [(\mathbf{new}\, x)([\![f]\!]_{out} \mid !\,\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\ & & \quad \mathbf{try} \\ & & \qquad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\, x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\ & & \quad \mathbf{catch}\ \mathbf{stop})]\end{array} \tag{A.46.1}$$

where in $[\![f]\!]_{out}$ all free occurrence of x in sub expressions h such that x immediately occurs in h (for instance h is $n.S(x)$) are encoded with a one time unfolding as follows

$$[\![h]\!]_{out} \triangleq [[\![x]\!]_{out_1} \mid \mathtt{in} \downarrow out_1(x).[\![h]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)] \tag{A.46.2}$$

Considering (A.46.2) we prove that

$$[(\mathtt{new}\, x)([\![f]\!]_{out} \mid !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid x \blacktriangleright [!\,\mathtt{out} \leftarrow val(c)])] \approx [\![f\{x{\leftarrow}c\}]\!]_{out} \tag{A.46.3}$$

For the case that f is a **where**, or a pipeline, or a parallel composition the result is direct by i.h. and the congruence property of $\approx$ (Theorem 5.3). The two remaining cases are site and expression call, which are analogous to prove so we show only site call. So we have that f is of the form $n.S(x)$. We have that

$$f\{x{\leftarrow}c\} = n.S(c) \tag{A.46.4}$$

In accordance with (A.46.2) we have

$$[\![f]\!]_{out} \triangleq [[\![x]\!]_{out_1} \mid \mathtt{in} \downarrow out_1(x).[\![f]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)] \tag{A.46.5}$$

From (A.46.3) and (A.46.5) and considering f is $n.S(x)$ we obtain, by expanding $[\![f]\!]_{out}$ and $[\![x]\!]_{out_1}$ and f, that

$$\begin{array}{ll} [(\mathtt{new}\, x) & ([x \blacktriangleleft [\mathtt{in} \leftarrow val(y).\mathtt{out} \uparrow out_1(y)] \mid \\ & \mathtt{in} \downarrow out_1(x).[\![n.S(x)]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)] \mid \\ & !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid x \blacktriangleright [!\,\mathtt{out} \leftarrow val(c)])] \end{array} \tag{A.46.6}$$

We then have that the only possible behavior of (A.46.6) is

$$\begin{array}{l} [(\mathtt{new}\, x) \quad ([x \blacktriangleleft [\mathtt{in} \leftarrow val(y).\mathtt{out} \uparrow out_1(y)] \mid \\ \qquad\quad \mathtt{in} \downarrow out_1(x).[\![n.S(x)]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)] \mid \\ \qquad\quad !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid x \blacktriangleright [!\,\mathtt{out} \leftarrow val(c)])] \\ \stackrel{\tau}{\Rightarrow} \\ [\,[[\![n.S(x)]\!]_{out_2}\{x{\leftarrow}c\} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)] \mid \\ !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid (\mathtt{new}\, x)(x \blacktriangleright [!\,\mathtt{out} \leftarrow val(c)])] \end{array} \tag{A.46.7}$$

Considering Lemma A.45 we have

$$\begin{array}{l} [\,[[\![n.S(x)]\!]_{out_2}\{x{\leftarrow}c\} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)] \mid \\ !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid (\mathtt{new}\, x)(x \blacktriangleright [!\,\mathtt{out} \leftarrow val(c)])] \\ \approx \\ [\,[[\![n.S(x)]\!]_{out_2}\{x{\leftarrow}c\} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)] \mid \\ !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x)] \end{array} \tag{A.46.8}$$

Considering Lemma A.42 we have

$$\begin{array}{l} [\,[[\![n.S(x)]\!]_{out_2}\{x{\leftarrow}c\} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x)] \mid \\ !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x)] \\ \approx \\ [\,[\![n.S(x)]\!]_{out}\{x{\leftarrow}c\} \mid !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x)] \end{array} \tag{A.46.9}$$

And again considering Lemma A.42 we have

$$\begin{array}{l} [\,[\![n.S(x)]\!]_{out}\{x{\leftarrow}c\} \mid !\,\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x)] \\ \approx \\ [\![n.S(x)]\!]_{out}\{x{\leftarrow}c\} \end{array} \tag{A.46.10}$$

By i.h. we have

$$[\![n.S(x)]\!]_{out}\{x{\leftarrow}c\} \approx [\![n.S(c)]\!] \tag{A.46.11}$$

which completes the proof for this case.

Proof of Lemma A.47

Lemma A.47 *Let O be an Orc process and D the definitions of the expressions used by O and $\widetilde{n}$ the names of the sites where the expressions are defined. We have*

$$
\begin{array}{lcl}
O \xrightarrow{\tau} O' & \Longrightarrow & (\mathbf{new}\ \widetilde{n})([\![O]\!]_{out} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![O']\!]_{out} \mid [\![D]\!]) \\
O \xrightarrow{!c} O' & \Longrightarrow & (\mathbf{new}\ \widetilde{n})([\![O]\!]_{out} \mid [\![D]\!]) \stackrel{\overline{\downarrow out(c)}}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![O']\!]_{out} \mid [\![D]\!]) \\
O \xrightarrow{p.S<c,u>} O' & \Longrightarrow & (\mathbf{new}\ \widetilde{n})([\![O]\!]_{out} \mid [\![D]\!]) \stackrel{\overline{(\nu u)p \blacktriangleright \mathtt{def}\ S}}{\Rightarrow} \stackrel{\overline{u \blacktriangleleft \leftarrow args(c)}}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![O']\!]_{out} \mid [\![D]\!]) \\
O \xrightarrow{u?c} O' & \Longrightarrow & (\mathbf{new}\ \widetilde{n})([\![O]\!]_{out} \mid [\![D]\!]) \stackrel{u \blacktriangleleft \leftarrow result(c)}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![O']\!]_{out} \mid [\![D]\!])
\end{array}
$$

Proof. We proceed by induction on the derivation of the transition labels. Throughout the proof we implicitly use the congruence property of $\approx$ (Theorem 5.3) and the result of Lemma A.44.

($Case\ f \gg x \gg g \xrightarrow{\tau} f' \gg x \gg g$)

We have

$$f \gg x \gg g \xrightarrow{\tau} f' \gg x \gg g \tag{A.47.1}$$

where (A.47.1) is derived from

$$f \xrightarrow{\tau} f' \tag{A.47.2}$$

We have that

$$[\![f \gg x \gg g]\!]_{out} \triangleq [[\![f]\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \tag{A.47.3}$$

By i.h. on (A.47.2) we conclude

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} [\![f']\!]_{out_1} \mid [\![D]\!] \tag{A.47.4}$$

From (A.47.3) and (A.47.4) we derive

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([\![f \gg x \gg g]\!]_{out} \mid [\![D]\!]) \\
\stackrel{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\ \widetilde{n})([[\![f']\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!])
\end{array} \tag{A.47.5}
$$

which completes the proof for this case.

($Case\ f \gg x \gg g \xrightarrow{\tau} (f' \gg x \gg g) \mid g\{x \leftarrow c\}$)

We have

$$f \gg x \gg g \xrightarrow{\tau} (f' \gg x \gg g) \mid g\{x \leftarrow c\} \tag{A.47.6}$$

where (A.47.6) is derived from

$$f \xrightarrow{!c} f' \tag{A.47.7}$$

We have that

$$[\![f \gg x \gg g]\!]_{out} \triangleq [[\![f]\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \tag{A.47.8}$$

By i.h. on A.47.7 we have

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \stackrel{\overline{\downarrow out_1(c)}}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.47.9}$$

From (A.47.8) and (A.47.9) we derive

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([\![f \gg x \gg g]\!]_{out} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\ \widetilde{n})([[\![f']\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)) \\
\qquad \mid [\![g\{x \leftarrow c\}]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)] \mid [\![D]\!])
\end{array} \tag{A.47.10}
$$

From (A.47.10) and Lemma A.43 we conclude

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})([\![f \gg x \gg g]\!]_{out} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} \\ (\mathbf{new}\,\widetilde{n})([[\![f']\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \\ \qquad \mid [\![g\{x \leftarrow c\}]\!]_{out} \mid \mid [\![D]\!])\end{array} \tag{A.47.11}$$

which completes the proof for this case.

(*Case* $f\ \mathbf{where}\ x :\in g \stackrel{\tau}{\longrightarrow} f\ \mathbf{where}\ x :\in g'$)

We have

$$f\ \mathbf{where}\ x :\in g \stackrel{\tau}{\longrightarrow} f\ \mathbf{where}\ x :\in g' \tag{A.47.12}$$

where (A.47.12) is derived from

$$g \stackrel{\tau}{\longrightarrow} g' \tag{A.47.13}$$

We have that

$$\begin{array}{lll}[\![f\ \mathbf{where}\ x :\in g]\!]_{out} & \triangleq & [(\mathbf{new}\,x)(\\ & & \quad [\![f]\!]_{out} \mid \\ & & \quad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\ & & \quad \mathbf{try} \\ & & \qquad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\ & & \quad \mathbf{catch}\,\mathbf{0})]\end{array} \tag{A.47.14}$$

By i.h. on A.47.13 we have

$$(\mathbf{new}\,\widetilde{n})([\![g]\!]_{out_2} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![g']\!]_{out_2} \mid [\![D]\!]) \tag{A.47.15}$$

From (A.47.14) and (A.47.15) we derive

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})([\![f\ \mathbf{where}\ x :\in g]\!]_{out} \mid [\![D]\!]) \\ \stackrel{\tau}{\Rightarrow}_{\approx} \\ (\mathbf{new}\,\widetilde{n})(\\ [\,(\mathbf{new}\,x)(\\ \quad [\![f]\!]_{out} \mid \\ \quad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\ \quad \mathbf{try} \\ \qquad [\![g']\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\ \quad \mathbf{catch}\,\mathbf{0})\,] \\ \mid [\![D]\!])\end{array} \tag{A.47.16}$$

which completes the proof for this case.

(*Case* $f\ \mathbf{where}\ x :\in g \stackrel{\tau}{\longrightarrow} f'\ \mathbf{where}\ x :\in g$)

We have

$$f\ \mathbf{where}\ x :\in g \stackrel{\tau}{\longrightarrow} f'\ \mathbf{where}\ x :\in g \tag{A.47.17}$$

where (A.47.17) is derived from

$$f \stackrel{\tau}{\longrightarrow} f' \tag{A.47.18}$$

We have that

$$\begin{array}{lll}[\![f\ \mathbf{where}\ x :\in g]\!]_{out} & \triangleq & [(\mathbf{new}\,x)(\\ & & \quad [\![f]\!]_{out} \mid \\ & & \quad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\ & & \quad \mathbf{try} \\ & & \qquad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\ & & \quad \mathbf{catch}\,\mathbf{0})]\end{array} \tag{A.47.19}$$

By i.h. on A.47.18 we have

$$(\mathbf{new}\,\widetilde{n})([\![f]\!]_{out} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![f']\!]_{out} \mid [\![D]\!]) \tag{A.47.20}$$

From (A.47.19) and (A.47.20) we derive

$$\begin{array}{l}
(\mathbf{new}\,\widetilde{n})([\![f\ \mathbf{where}\ x :\in g]\!]_{out} \mid [\![D]\!]) \\
\stackrel{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\,\widetilde{n})(\\
[\,(\mathbf{new}\,x)(\\
\quad [\![f']\!]_{out} \mid \\
\quad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\quad \mathbf{try} \\
\quad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\quad \mathbf{catch}\,\mathbf{0})\,] \\
\mid [\![D]\!])
\end{array} \tag{A.47.21}$$

which completes the proof for this case.

(*Case* $f\ \mathbf{where}\ x :\in g \stackrel{\tau}{\longrightarrow} f\{x \leftarrow c\}$)

We have

$$f\ \mathbf{where}\ x :\in g \stackrel{\tau}{\longrightarrow} f\{x \leftarrow c\} \tag{A.47.22}$$

where (A.47.22) is derived from

$$g \stackrel{!c}{\longrightarrow} g' \tag{A.47.23}$$

We have that

$$\begin{array}{rcl}
[\![f\ \mathbf{where}\ x :\in g]\!]_{out} & \triangleq & [(\mathbf{new}\,x)(\\
& & \quad [\![f]\!]_{out} \mid \\
& & \quad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
& & \quad \mathbf{try} \\
& & \quad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
& & \quad \mathbf{catch}\,\mathbf{0})]
\end{array} \tag{A.47.24}$$

By i.h. on A.47.23 we have

$$(\mathbf{new}\,\widetilde{n})([\![g]\!]_{out_2} \mid [\![D]\!]) \stackrel{\overline{\downarrow out_2(c)}}{\Rightarrow} \approx (\mathbf{new}\,\widetilde{n})([\![g']\!]_{out_2} \mid [\![D]\!]) \tag{A.47.25}$$

From (A.47.24) and (A.47.25) we derive

$$\begin{array}{l}
(\mathbf{new}\,\widetilde{n})([\![f\ \mathbf{where}\ x :\in g]\!]_{out} \mid [\![D]\!]) \\
\stackrel{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\,\widetilde{n})(\\
[\,(\mathbf{new}\,x)(\\
\quad [\![f]\!]_{out} \mid \\
\quad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\quad \mathbf{try} \\
\quad\quad [\![g']\!]_{out_2} \mid \mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(c)] \\
\quad \mathbf{catch}\,\mathbf{0})\,] \\
\mid [\![D]\!])
\end{array} \tag{A.47.26}$$

From (A.47.26) we derive

$$(\mathbf{new}\,\widetilde{n})([\![f\ \mathbf{where}\ x :\in g]\!]_{out} \mid [\![D]\!]) \quad \stackrel{\tau}{\Rightarrow}_{\approx} \quad (\mathbf{new}\,\widetilde{n})([\ (\mathbf{new}\,x)(\\ [\![f]\!]_{out} \mid \\ !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(c)])\ ] \mid [\![D]\!]) \tag{A.47.27}$$

From (A.47.27) and considering Lemma A.46 we conclude

$$(\mathbf{new}\,\widetilde{n})([\![f\ \mathbf{where}\ x :\in g]\!]_{out} \mid [\![D]\!]) \quad \stackrel{\tau}{\Rightarrow}_{\approx} \quad (\mathbf{new}\,\widetilde{n})([\![f\{x \leftarrow c\}]\!] \mid [\![D]\!]) \tag{A.47.28}$$

which completes the proof for this case.

($Case\ f \mid g \stackrel{\lambda}{\longrightarrow} f' \mid g$)

We have

$$f \mid g \stackrel{\lambda}{\longrightarrow} f' \mid g \tag{A.47.29}$$

where (A.47.29) is derived from

$$f \stackrel{\lambda}{\longrightarrow} f' \tag{A.47.30}$$

We have that

$$[\![f \mid g]\!]_{out} \triangleq [\![f]\!]_{out} \mid [\![g]\!]_{out} \tag{A.47.31}$$

By i.h. on (A.47.30) we conclude

$$(\mathbf{new}\,\widetilde{n})([\![f]\!]_{out} \mid [\![D]\!]) \stackrel{\lambda}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![f']\!]_{out} \mid [\![D]\!]) \tag{A.47.32}$$

From (A.47.31) and (A.47.32) we derive

$$(\mathbf{new}\,\widetilde{n})([\![f \mid g]\!]_{out} \mid [\![D]\!]) \stackrel{\lambda}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![f']\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!]) \tag{A.47.33}$$

which completes the proof for this case.

($Case\ f \mid g \stackrel{\lambda}{\longrightarrow} f \mid g'$)

We have

$$f \mid g \stackrel{\lambda}{\longrightarrow} f \mid g' \tag{A.47.34}$$

where (A.47.34) is derived from

$$g \stackrel{\lambda}{\longrightarrow} g' \tag{A.47.35}$$

We have that

$$[\![f \mid g]\!]_{out} \triangleq [\![f]\!]_{out} \mid [\![g]\!]_{out} \tag{A.47.36}$$

By i.h. on (A.47.35) we conclude

$$(\mathbf{new}\,\widetilde{n})([\![g]\!]_{out} \mid [\![D]\!]) \stackrel{\lambda}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![g']\!]_{out} \mid [\![D]\!]) \tag{A.47.37}$$

From (A.47.36) and (A.47.37) we derive

$$(\mathbf{new}\,\widetilde{n})([\![f \mid g]\!]_{out} \mid [\![D]\!]) \stackrel{\lambda}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![f]\!]_{out} \mid [\![g']\!]_{out} \mid [\![D]\!]) \tag{A.47.38}$$

which completes the proof for this case.

($Case\ let(c) \stackrel{!c}{\longrightarrow} 0$)

We have

$$let(c) \stackrel{!c}{\longrightarrow} 0 \tag{A.47.39}$$

We have that

$$[\![let(c)]\!]_{out} \triangleq \mathtt{out} \downarrow out(c) \tag{A.47.40}$$

From (A.47.40) we conclude

$$(\mathtt{new}\ \widetilde{n})([\![let(c)]\!]_{out} \mid [\![D]\!]) \overset{\overline{\downarrow out(c)}}{\Rightarrow} \approx (\mathtt{new}\ \widetilde{n})([\![0]\!]_{out} \mid [\![D]\!]) \tag{A.47.41}$$

which completes the proof for this case.

(*Case* $?u \xrightarrow{u?c} let(c)$)

We have

$$?u \xrightarrow{u?c} let(c) \tag{A.47.42}$$

We have that

$$[\![?u]\!]_{out} \triangleq u \blacktriangleleft [\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)] \tag{A.47.43}$$

We derive that

$$(\mathtt{new}\ \widetilde{n})([\![?u]\!]_{out} \mid [\![D]\!]) \overset{u\blacktriangleleft\leftarrow result(c)}{\Rightarrow} \approx (\mathtt{new}\ \widetilde{n})(u \blacktriangleleft [\mathtt{out} \uparrow out(c)] \mid [\![D]\!]) \tag{A.47.44}$$

From (A.47.44) and considering Proposition 5.4(5) and 5.4(4) we conclude

$$(\mathtt{new}\ \widetilde{n})([\![?u]\!]_{out} \mid [\![D]\!]) \overset{u\blacktriangleleft\leftarrow result(c)}{\Rightarrow} \approx (\mathtt{new}\ \widetilde{n})(\mathtt{out} \downarrow out(c) \mid [\![D]\!]) \tag{A.47.45}$$

which completes the proof for this case.

(*Case* $p.S(c) \xrightarrow{p.S<c,u>} ?u$)

We have

$$p.S(c) \xrightarrow{p.S<c,u>} ?u \tag{A.47.46}$$

where u is fresh. We have that

$$[\![p.S(c)]\!]_{out} \triangleq \mathtt{instance}\ p \blacktriangleright S \Leftarrow (\mathtt{out} \leftarrow args(c).\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)) \tag{A.47.47}$$

Considering (A.47.47) we derive

$$\begin{array}{l}(\mathtt{new}\ \widetilde{n})([\![p.S(c)]\!] \mid [\![D]\!]) \\ \overset{\overline{(\nu u)p\blacktriangleright def\ S}}{\Rightarrow} \approx \\ (\mathtt{new}\ \widetilde{n})(u \blacktriangleleft [\mathtt{out} \leftarrow args(c).\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)] \mid [\![D]\!]) \\ \overset{\overline{u\blacktriangleleft\leftarrow args(c)}}{\Rightarrow} \approx \\ (\mathtt{new}\ \widetilde{n})(u \blacktriangleleft [\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)] \mid [\![D]\!])\end{array} \tag{A.47.48}$$

which completes the proof for this case.

(*Case* $n.S(c) \xrightarrow{\tau} e\{x \leftarrow c\}$)

We have

$$n.S(c) \xrightarrow{\tau} e\{x \leftarrow c\} \tag{A.47.49}$$

We have that

$$[\![p.S(c)]\!]_{out} \triangleq \mathtt{instance}\ n \blacktriangleright S \Leftarrow (\mathtt{out} \leftarrow args(c).!\ \mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)) \tag{A.47.50}$$

We have that in D there is an association between $n.S(x)$ and e hence in $[\![D]\!]$ there is $[\![n.S(x) = e]\!]$ which is encoded as follows

$$[\![n.S(x) = e]\!] \triangleq n \blacktriangleright [!\ \mathtt{def}\ S \Rightarrow (\mathtt{in} \leftarrow args(x).[\![e]\!]_{out} \mid !\ \mathtt{in} \downarrow out(x).\mathtt{out} \leftarrow result(x))] \tag{A.47.51}$$

and we have that $n \in \widetilde{n}$. We derive that

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([\![n.S(c)]\!]_{out} \mid [\![D]\!]) \\
\stackrel{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\ \widetilde{n}) \\
\quad ((\mathbf{new}\ a)(a \blacktriangleleft [!\ \mathbf{in} \leftarrow result(x).\mathbf{out} \uparrow out(x)] \\
\qquad\qquad \mid a \blacktriangleright [[\![e]\!]_{out}\{x \leftarrow c\} \mid !\ \mathbf{in} \downarrow out(x).\mathbf{out} \leftarrow result(x)]) \\
\quad \mid [\![D]\!])
\end{array}
\tag{A.47.52}
$$

From (A.47.52) and considering Lemma A.44 we have

$$
(\mathbf{new}\ \widetilde{n})([\![n.S(c)]\!]_{out} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![e]\!]_{out}\{x \leftarrow c\} \mid [\![D]\!]) \tag{A.47.53}
$$

which completes the proof for this last case.

Proof of Lemma A.48

Lemma A.48 *Let O be an Orc process and D the definitions of the expressions used by O and $\widetilde{n}$ the names of the sites where the expressions are defined. We have*

$$
\begin{array}{ll}
(\mathbf{new}\ \widetilde{n})([\![O]\!]_{out} \mid [\![D]\!]) \stackrel{\tau}{\longrightarrow} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) & \Longrightarrow O \stackrel{\tau}{\longrightarrow} O' \wedge \\
 & (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![O']\!]_{out} \mid [\![D]\!]) \\
(\mathbf{new}\ \widetilde{n})([\![O]\!]_{out} \mid [\![D]\!]) \stackrel{\overline{\downarrow out(c)}}{\longrightarrow} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) & \Longrightarrow O \stackrel{!c}{\longrightarrow} O' \wedge \\
 & (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![O']\!]_{out} \mid [\![D]\!]) \\
(\mathbf{new}\ \widetilde{n})([\![O]\!]_{out} \mid [\![D]\!]) \stackrel{\overline{(\nu u)p \blacktriangleright def\ S}}{\longrightarrow} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) & \Longrightarrow O \stackrel{p.S<c,u>}{\longrightarrow} O' \wedge \\
 & (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \stackrel{\overline{u \blacktriangleleft \leftarrow args(c)}}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![O']\!]_{out} \mid [\![D]\!]) \\
(\mathbf{new}\ \widetilde{n})([\![O]\!]_{out} \mid [\![D]\!]) \stackrel{u \blacktriangleleft \leftarrow result(c)}{\longrightarrow} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) & \Longrightarrow O \stackrel{u?c}{\longrightarrow} O' \wedge \\
 & (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![O']\!]_{out} \mid [\![D]\!])
\end{array}
$$

Proof. We proceed by induction on the definition of the encoding.

(**Case** $[\![f \gg x \gg g]\!]_{out}$)

We consider the possible transitions of

$$
(\mathbf{new}\ \widetilde{n})([[\![f]\!]_{out_1} \mid !\ \mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\ \mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!]) \tag{A.48.1}
$$

(*Case* $[\![f]\!]_{out_1} \stackrel{\tau}{\longrightarrow} P$)

We have

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([[\![f]\!]_{out_1} \mid !\ \mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\ \mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!]) \\
\stackrel{\tau}{\longrightarrow} \\
(\mathbf{new}\ \widetilde{n})([P \mid !\ \mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\ \mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!])
\end{array}
\tag{A.48.2}
$$

where (A.48.2) is derived from

$$
(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \stackrel{\tau}{\longrightarrow} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.3}
$$

By i.h. on (A.48.3) we have that

$$
f \stackrel{\tau}{\longrightarrow} f' \tag{A.48.4}
$$

and

$$
(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.48.5}
$$

From (A.48.4) we derive

$$f \gg x \gg g \xrightarrow{\tau} f' \gg x \gg g \tag{A.48.6}$$

From (A.48.5) we derive

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})([P \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!]) \\ \stackrel{\tau}{\Rightarrow}_{\approx} \\ (\mathbf{new}\,\widetilde{n})([[\![f']\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!])\end{array} \tag{A.48.7}$$

which completes the proof for this case.

(*Case* $[\![f]\!]_{out_1} \xrightarrow{\overline{\downarrow out(c)}} P$)
We have

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})([[\![f]\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!]) \\ \xrightarrow{\tau} \\ (\mathbf{new}\,\widetilde{n})([P \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)) \mid \\ \qquad [\![g]\!]_{out_2}\{x \leftarrow c\} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)] \mid [\![D]\!])\end{array} \tag{A.48.8}$$

where (A.48.8) is derived from

$$(\mathbf{new}\,\widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \xrightarrow{\overline{\downarrow out(c)}} (\mathbf{new}\,\widetilde{n})(P \mid [\![D]\!]) \tag{A.48.9}$$

By i.h. on (A.48.9) we have that

$$f \xrightarrow{!c} f' \tag{A.48.10}$$

and

$$(\mathbf{new}\,\widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.48.11}$$

From (A.48.10) we derive

$$f \gg x \gg g \xrightarrow{\tau} (f' \gg x \gg g) \mid (g\{x \leftarrow c\}) \tag{A.48.12}$$

From (A.48.11) we derive

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})([P \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)) \mid \\ \qquad [\![g]\!]_{out_2}\{x \leftarrow c\} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)] \mid [\![D]\!]) \\ \stackrel{\tau}{\Rightarrow}_{\approx} \\ (\mathbf{new}\,\widetilde{n})([[\![f']\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)) \mid \\ \qquad [\![g]\!]_{out_2}\{x \leftarrow c\} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)] \mid [\![D]\!])\end{array} \tag{A.48.13}$$

From (A.48.13) and considering Lemma A.43 we conclude

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})([P \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)) \mid \\ \qquad [\![g]\!]_{out_2}\{x \leftarrow c\} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x)] \mid [\![D]\!]) \\ \stackrel{\tau}{\Rightarrow}_{\approx} \\ (\mathbf{new}\,\widetilde{n})([[\![f']\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid \\ \qquad [\![g]\!]_{out}\{x \leftarrow c\} \mid [\![D]\!])\end{array} \tag{A.48.14}$$

which completes the proof for this case.

(*Case* $[\![f]\!]_{out_1} \xrightarrow{\overline{(\nu u)p \blacktriangleright def\ S}} P$)
We have

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})([[\![f]\!]_{out_1} \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!]) \\ \xrightarrow{\overline{(\nu u)p \blacktriangleright def\ S}} \\ (\mathbf{new}\,\widetilde{n})([P \mid !\,\mathbf{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathbf{in} \downarrow out_2(x).\mathbf{out} \uparrow out(x))] \mid [\![D]\!])\end{array} \tag{A.48.15}$$

where (A.48.15) is derived from

$$(\mathtt{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \xrightarrow{\overline{(\nu u)p\blacktriangleright def\ S}} (\mathtt{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.16}$$

By i.h. on (A.48.16) we have that

$$f \xrightarrow{p.S<c,u>} f' \tag{A.48.17}$$

and

$$(\mathtt{new}\ \widetilde{n})(P \mid [\![D]\!]) \overset{\overline{(\nu u)\blacktriangleleft\leftarrow args(c)}}{\Rightarrow}_{\approx} (\mathtt{new}\ \widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.48.18}$$

From (A.48.17) we derive

$$f \gg x \gg g \xrightarrow{p.S<c,u>} f' \gg x \gg g \tag{A.48.19}$$

From (A.48.18) we derive

$$\begin{array}{l}(\mathtt{new}\ \widetilde{n})([P \mid !\,\mathtt{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x))] \mid [\![D]\!]) \\ \overset{\overline{(\nu u)\blacktriangleleft\leftarrow args(c)}}{\Rightarrow}_{\approx} \\ (\mathtt{new}\ \widetilde{n})([[\![f']\!]_{out_1} \mid !\,\mathtt{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x))] \mid [\![D]\!])\end{array} \tag{A.48.20}$$

which completes the proof for this case.

($Case\ [\![f]\!]_{out_1} \xrightarrow{u\blacktriangleleft\leftarrow result(c)} P$)

We have

$$\begin{array}{l}(\mathtt{new}\ \widetilde{n})([[\![f]\!]_{out_1} \mid !\,\mathtt{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x))] \mid [\![D]\!]) \\ \xrightarrow{u\blacktriangleleft\leftarrow result(c)} \\ (\mathtt{new}\ \widetilde{n})([P \mid !\,\mathtt{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x))] \mid [\![D]\!])\end{array} \tag{A.48.21}$$

where (A.48.21) is derived from

$$(\mathtt{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \xrightarrow{u\blacktriangleleft\leftarrow result(c)} (\mathtt{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.22}$$

By i.h. on (A.48.22) we have that

$$f \xrightarrow{u?c} f' \tag{A.48.23}$$

and

$$(\mathtt{new}\ \widetilde{n})(P \mid [\![D]\!]) \overset{\tau}{\Rightarrow}_{\approx} (\mathtt{new}\ \widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.48.24}$$

From (A.48.23) we derive

$$f \gg x \gg g \xrightarrow{u?c} f' \gg x \gg g \tag{A.48.25}$$

From (A.48.24) we derive

$$\begin{array}{l}(\mathtt{new}\ \widetilde{n})([P \mid !\,\mathtt{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x))] \mid [\![D]\!]) \\ \overset{\tau}{\Rightarrow}_{\approx} \\ (\mathtt{new}\ \widetilde{n})([[\![f']\!]_{out_1} \mid !\,\mathtt{in} \downarrow out_1(x).([\![g]\!]_{out_2} \mid !\,\mathtt{in} \downarrow out_2(x).\mathtt{out} \uparrow out(x))] \mid [\![D]\!])\end{array} \tag{A.48.26}$$

which completes the proof for this case.

(**Case** $[\![f \mid g]\!]_{out}$)

We consider the possible transitions of

$$(\mathtt{new}\ \widetilde{n})([\![f]\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!]) \tag{A.48.27}$$

that result from observations over $[\![f]\!]$ being the case for observations over $[\![g]\!]$ analogous to prove.

($Case\ [\![f]\!]_{out} \xrightarrow{\tau} P$)
We have

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!]) \\ \xrightarrow{\tau} \\ (\mathbf{new}\ \widetilde{n})(P \mid [\![g]\!]_{out} \mid [\![D]\!])\end{array} \tag{A.48.28}$$

where (A.48.28) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \xrightarrow{\tau} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.29}$$

By i.h. on (A.48.29) we have that

$$f \xrightarrow{\tau} f' \tag{A.48.30}$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.48.31}$$

From (A.48.30) we derive

$$f \mid g \xrightarrow{\tau} f' \mid g \tag{A.48.32}$$

From (A.48.31) we derive

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})(P \mid [\![g]\!]_{out} \mid [\![D]\!]) \\ \stackrel{\tau}{\Rightarrow}_{\approx} \\ (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!])\end{array} \tag{A.48.33}$$

which completes the proof for this case.

($Case\ [\![f]\!]_{out} \xrightarrow{\overline{\downarrow out(c)}} P$)
We have

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!]) \\ \xrightarrow{\overline{\downarrow out(c)}} \\ (\mathbf{new}\ \widetilde{n})(P \mid [\![g]\!]_{out} \mid [\![D]\!])\end{array} \tag{A.48.34}$$

where (A.48.34) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \xrightarrow{\overline{\downarrow out(c)}} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.35}$$

By i.h. on (A.48.35) we have that

$$f \xrightarrow{!c} f' \tag{A.48.36}$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.48.37}$$

From (A.48.36) we derive

$$f \mid g \xrightarrow{!c} f' \mid g \tag{A.48.38}$$

From (A.48.37) we derive

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})(P \mid [\![g]\!]_{out} \mid [\![D]\!]) \\ \stackrel{\tau}{\Rightarrow}_{\approx} \\ (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!])\end{array} \tag{A.48.39}$$

which completes the proof for this case.

($Case\ [\![f]\!]_{out} \xrightarrow{\overline{(\nu u)p \blacktriangleright def\ S}} P$)
We have

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!]) \\ \xrightarrow{\overline{(\nu u)p \blacktriangleright def\ S}} \\ (\mathbf{new}\ \widetilde{n})(P \mid [\![g]\!]_{out} \mid [\![D]\!])\end{array} \tag{A.48.40}$$

where (A.48.40) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \xrightarrow{\overline{(\nu u)p\blacktriangleright def\ S}} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.41}$$

By i.h. on (A.48.41) we have that

$$f \xrightarrow{p.S<c,u>} f' \tag{A.48.42}$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \overset{\overline{u\blacktriangleleft args(c)}}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.48.43}$$

From (A.48.42) we derive

$$f \mid g \xrightarrow{p.S<c,u>} f' \mid g \tag{A.48.44}$$

From (A.48.43) we derive

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})(P \mid [\![g]\!]_{out} \mid [\![D]\!]) \\ \overset{\overline{u\blacktriangleleft args(c)}}{\Rightarrow}_{\approx} \\ (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!])\end{array} \tag{A.48.45}$$

which completes the proof for this case.

(*Case* $[\![f]\!]_{out} \xrightarrow{u\blacktriangleleft\leftarrow result(c)} P$)

We have

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!]) \\ \xrightarrow{u\blacktriangleleft\leftarrow result(c)} \\ (\mathbf{new}\ \widetilde{n})(P \mid [\![g]\!]_{out} \mid [\![D]\!])\end{array} \tag{A.48.46}$$

where (A.48.46) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out_1} \mid [\![D]\!]) \xrightarrow{u\blacktriangleleft\leftarrow result(c)} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.47}$$

By i.h. on (A.48.47) we have that

$$f \xrightarrow{u?c} f' \tag{A.48.48}$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \overset{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out_1} \mid [\![D]\!]) \tag{A.48.49}$$

From (A.48.48) we derive

$$f \mid g \xrightarrow{u?c} f' \mid g \tag{A.48.50}$$

From (A.48.49) we derive

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})(P \mid [\![g]\!]_{out} \mid [\![D]\!]) \\ \overset{\tau}{\Rightarrow}_{\approx} \\ (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out} \mid [\![g]\!]_{out} \mid [\![D]\!])\end{array} \tag{A.48.51}$$

which completes the proof for this case.

(**Case** $[\![f\ \mathbf{where}\ x :\in g]\!]_{out}$)

We consider the possible transitions of

$$\begin{array}{l}(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\ \qquad [\![f]\!]_{out} \mid \\ \qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\ \qquad \mathbf{try} \\ \qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\ \qquad \mathbf{catch}\ \mathbf{0})] \\ \quad \mid [\![D]\!])\end{array} \tag{A.48.52}$$

(*Case* $[\![f]\!]_{out} \stackrel{\tau}{\Rightarrow} P$)
We have

$$
\begin{array}{l}
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\, x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\, \mathbf{0})] \\
\quad \mid [\![D]\!]) \\
\stackrel{\tau}{\Rightarrow} \\
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad P \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\, x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\, \mathbf{0})] \\
\quad \mid [\![D]\!])
\end{array}
\tag{A.48.53}
$$

where (A.48.53) is derived from

$$(\mathbf{new}\,\widetilde{n})([\![f]\!]_{out} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow} (\mathbf{new}\,\widetilde{n})(P \mid [\![D]\!]) \tag{A.48.54}$$

By i.h. on (A.48.54) we have

$$f \stackrel{\tau}{\longrightarrow} f' \tag{A.48.55}$$

and

$$(\mathbf{new}\,\widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![f']\!]_{out} \mid [\![D]\!]) \tag{A.48.56}$$

From (A.48.55) we derive

$$f\ \mathbf{where}\ x :\in g \stackrel{\tau}{\longrightarrow} f'\ \mathbf{where}\ x :\in g \tag{A.48.57}$$

From (A.48.56) we conclude

$$
\begin{array}{l}
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad P \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\, x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\, \mathbf{0})] \\
\quad \mid [\![D]\!]) \\
\stackrel{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad [\![f']\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\, x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\, \mathbf{0})] \\
\quad \mid [\![D]\!])
\end{array}
\tag{A.48.58}
$$

which completes the proof for this case.

($Case\ [\![f]\!]_{out} \overset{\overline{\downarrow out(c)}}{\Rightarrow} P$)
We have

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!]) \\
\overset{\overline{\downarrow out(c)}}{\Rightarrow} \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad P \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!])
\end{array}
\tag{A.48.59}
$$

where (A.48.59) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out} \mid [\![D]\!]) \overset{\overline{\downarrow out(c)}}{\Rightarrow} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.60}$$

By i.h. on (A.48.60) we have

$$f \xrightarrow{!c} f' \tag{A.48.61}$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \overset{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out} \mid [\![D]\!]) \tag{A.48.62}$$

From (A.48.61) we derive

$$f\ \mathbf{where}\ x :\in g \xrightarrow{!c} f'\ \mathbf{where}\ x :\in g \tag{A.48.63}$$

From (A.48.62) we conclude

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad P \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!]) \\
\overset{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f']\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!])
\end{array}
\tag{A.48.64}
$$

which completes the proof for this case.

$(Case\ [\![f]\!]_{out}\ \overset{\overline{(\nu u)p\blacktriangleright def\ S}}{\Rightarrow}\ P)$
We have

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\qquad \mid [\![D]\!]) \\
\overset{\overline{(\nu u)p\blacktriangleright def\ S}}{\Rightarrow} \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad P \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\qquad \mid [\![D]\!])
\end{array}
\tag{A.48.65}
$$

where (A.48.65) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out} \mid [\![D]\!])\ \overset{\overline{(\nu u)p\blacktriangleright def\ S}}{\Rightarrow}\ (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.66}$$

By i.h. on (A.48.66) we have

$$f\ \overset{p.S<c,u>}{\longrightarrow}\ f' \tag{A.48.67}$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!])\ \overset{\overline{u\blacktriangleleft\leftarrow args(c)}}{\Rightarrow}_{\approx}\ (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out} \mid [\![D]\!]) \tag{A.48.68}$$

From (A.48.67) we derive

$$f\ \mathbf{where}\ x :\in g\ \overset{p.S<c,u>}{\longrightarrow}\ f'\ \mathbf{where}\ x :\in g \tag{A.48.69}$$

From (A.48.68) we conclude

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad P \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\qquad \mid [\![D]\!]) \\
\overset{\overline{u\blacktriangleleft\leftarrow args(c)}}{\Rightarrow}_{\approx} \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f']\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\qquad \mid [\![D]\!])
\end{array}
\tag{A.48.70}
$$

which completes the proof for this case.

($Case\ [\![f]\!]_{out} \overset{u\blacktriangleleft\leftarrow result(c)}{\Rightarrow} P$)

We have

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!]) \\
\overset{u\blacktriangleleft\leftarrow result(c)}{\Rightarrow} \qquad\qquad (\text{A.48.71}) \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad P \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!])
\end{array}
$$

where (A.48.71) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![f]\!]_{out} \mid [\![D]\!]) \overset{u\blacktriangleleft\leftarrow result(c)}{\Rightarrow} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \qquad (\text{A.48.72})$$

By i.h. on (A.48.72) we have

$$f \xrightarrow{u?c} f' \qquad (\text{A.48.73})$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \overset{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![f']\!]_{out} \mid [\![D]\!]) \qquad (\text{A.48.74})$$

From (A.48.73) we derive

$$f\ \mathbf{where}\ x :\in g \xrightarrow{u?c} f'\ \mathbf{where}\ x :\in g \qquad (\text{A.48.75})$$

From (A.48.74) we conclude

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad P \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!]) \\
\overset{\tau}{\Rightarrow}_{\approx} \qquad\qquad (\text{A.48.76}) \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f']\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!])
\end{array}
$$

which completes the proof for this case.

(*Case* $[\![g]\!]_{out_2} \stackrel{\tau}{\Rightarrow} P$)

We have

$$
\begin{array}{l}
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad\qquad [\![f]\!]_{out} \mid \\
\qquad\qquad !\mathbf{in}\downarrow out(x).\mathbf{out}\uparrow out(x) \mid \\
\qquad\qquad \mathbf{try} \\
\qquad\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in}\downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\qquad\qquad \mathbf{catch}\,\mathbf{0})] \\
\qquad \mid [\![D]\!]) \\
\stackrel{\tau}{\Rightarrow} \\
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad\qquad [\![f]\!]_{out} \mid \\
\qquad\qquad !\mathbf{in}\downarrow out(x).\mathbf{out}\uparrow out(x) \mid \\
\qquad\qquad \mathbf{try} \\
\qquad\qquad\quad P \mid \mathbf{in}\downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\qquad\qquad \mathbf{catch}\,\mathbf{0})] \\
\qquad \mid [\![D]\!])
\end{array}
\tag{A.48.77}
$$

where (A.48.77) is derived from

$$(\mathbf{new}\,\widetilde{n})([\![g]\!]_{out_2} \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow} (\mathbf{new}\,\widetilde{n})(P \mid [\![D]\!]) \tag{A.48.78}$$

By i.h. on (A.48.78) we have

$$g \xrightarrow{\tau} g' \tag{A.48.79}$$

and

$$(\mathbf{new}\,\widetilde{n})(P \mid [\![D]\!]) \stackrel{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\,\widetilde{n})([\![g']\!]_{out_2} \mid [\![D]\!]) \tag{A.48.80}$$

From (A.48.79) we derive

$$f\;\mathbf{where}\;x :\in g \xrightarrow{\tau} f\;\mathbf{where}\;x :\in g' \tag{A.48.81}$$

From (A.48.80) we conclude

$$
\begin{array}{l}
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad\qquad [\![f]\!]_{out} \mid \\
\qquad\qquad !\mathbf{in}\downarrow out(x).\mathbf{out}\uparrow out(x) \mid \\
\qquad\qquad \mathbf{try} \\
\qquad\qquad\quad P \mid \mathbf{in}\downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\qquad\qquad \mathbf{catch}\,\mathbf{0})] \\
\qquad \mid [\![D]\!]) \\
\stackrel{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad\qquad [\![f']\!]_{out} \mid \\
\qquad\qquad !\mathbf{in}\downarrow out(x).\mathbf{out}\uparrow out(x) \mid \\
\qquad\qquad \mathbf{try} \\
\qquad\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in}\downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathbf{out} \leftarrow val(y)] \\
\qquad\qquad \mathbf{catch}\,\mathbf{0})] \\
\qquad \mid [\![D]\!])
\end{array}
\tag{A.48.82}
$$

which completes the proof for this case.

($Case\ [\![g]\!]_{out_2} \overset{\overline{\downarrow out_2(c)}}{\Rightarrow} P$)
We have

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!]) \qquad\qquad (\text{A.48.83}) \\
\overset{\tau}{\Rightarrow} \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad x \blacktriangleright [!\ \mathbf{out} \leftarrow val(c)])] \\
\quad \mid [\![D]\!])
\end{array}
$$

where (A.48.83) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![g]\!]_{out_2} \mid [\![D]\!]) \overset{\overline{\downarrow out_2(c)}}{\Rightarrow} (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \qquad (\text{A.48.84})$$

By i.h. on (A.48.84) we have

$$g \xrightarrow{!c} g' \qquad (\text{A.48.85})$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \overset{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![g']\!]_{out_2} \mid [\![D]\!]) \qquad (\text{A.48.86})$$

From (A.48.85) we derive

$$f\ \mathbf{where}\ x :\in g \xrightarrow{\tau} f\{x \leftarrow c\} \qquad (\text{A.48.87})$$

From Lemma A.46 we conclude

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad x \blacktriangleright [!\ \mathbf{out} \leftarrow val(c)])] \\
\quad \mid [\![D]\!]) \\
\approx (\mathbf{new}\ \widetilde{n})([\![f]\!]_{out}\{x \leftarrow c\} \mid [\![D]\!])
\end{array}
\qquad (\text{A.48.88})
$$

which completes the proof for this case.
($Case\ [\![g]\!]_{out_2} \overset{\overline{(\nu u)p \blacktriangleright def\ S}}{\Rightarrow} P$)

We have

$$
\begin{array}{l}
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathtt{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathtt{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\,\mathbf{0})] \\
\qquad\quad \mid [\![D]\!]) \\
\xrightarrow[\Rightarrow]{\overline{(\nu u)p\blacktriangleright def\ S}} \\
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad P \mid \mathtt{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathtt{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\,\mathbf{0})] \\
\qquad\quad \mid [\![D]\!])
\end{array}
\tag{A.48.89}
$$

where (A.48.89) is derived from

$$
(\mathbf{new}\,\widetilde{n})([\![g]\!]_{out_2} \mid [\![D]\!]) \overset{\overline{(\nu u)p\blacktriangleright def\ S}}{\Rightarrow} (\mathbf{new}\,\widetilde{n})(P \mid [\![D]\!]) \tag{A.48.90}
$$

By i.h. on (A.48.90) we have

$$
g \overset{p.S<c,u>}{\longrightarrow} g' \tag{A.48.91}
$$

and

$$
(\mathbf{new}\,\widetilde{n})(P \mid [\![D]\!]) \overset{\overline{u\blacktriangleleft\leftarrow args(c)}}{\Rightarrow} \approx (\mathbf{new}\,\widetilde{n})([\![g']\!]_{out_2} \mid [\![D]\!]) \tag{A.48.92}
$$

From (A.48.91) we derive

$$
f\ \mathbf{where}\ x :\in g \overset{p.S<c,u>}{\longrightarrow} f\ \mathbf{where}\ x :\in g' \tag{A.48.93}
$$

From (A.48.92) we conclude

$$
\begin{array}{l}
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad P \mid \mathtt{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathtt{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\,\mathbf{0})] \\
\qquad\quad \mid [\![D]\!]) \\
\xrightarrow[\Rightarrow]{\overline{u\blacktriangleleft\leftarrow args(c)}} \approx \\
(\mathbf{new}\,\widetilde{n})([(\mathbf{new}\,x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathtt{in} \downarrow out(x).\mathtt{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g']\!]_{out_2} \mid \mathtt{in} \downarrow out_2(y).\mathbf{throw}\,x \blacktriangleright [!\,\mathtt{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\,\mathbf{0})] \\
\qquad\quad \mid [\![D]\!])
\end{array}
\tag{A.48.94}
$$

which completes the proof for this case.

$(Case\ [\![g]\!]_{out_2}\ \overset{u\blacktriangleleft\leftarrow result(c)}{\Rightarrow}\ P)$
We have

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g]\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!]) \\
\overset{u\blacktriangleleft\leftarrow result(c)}{\Rightarrow} \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad P \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!])
\end{array}
\tag{A.48.95}
$$

where (A.48.95) is derived from

$$(\mathbf{new}\ \widetilde{n})([\![g]\!]_{out_2} \mid [\![D]\!])\ \overset{u\blacktriangleleft\leftarrow result(c)}{\Rightarrow}\ (\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \tag{A.48.96}$$

By i.h. on (A.48.96) we have

$$g \xrightarrow{u?c} g' \tag{A.48.97}$$

and

$$(\mathbf{new}\ \widetilde{n})(P \mid [\![D]\!]) \overset{\tau}{\Rightarrow}_{\approx} (\mathbf{new}\ \widetilde{n})([\![g']\!]_{out_2} \mid [\![D]\!]) \tag{A.48.98}$$

From (A.48.97) we derive

$$f\ \mathbf{where}\ x :\in g \xrightarrow{u?c} f\ \mathbf{where}\ x :\in g' \tag{A.48.99}$$

From (A.48.98) we conclude

$$
\begin{array}{l}
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad P \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!]) \\
\overset{\tau}{\Rightarrow}_{\approx} \\
(\mathbf{new}\ \widetilde{n})([(\mathbf{new}\ x)(\\
\qquad [\![f]\!]_{out} \mid \\
\qquad !\mathbf{in} \downarrow out(x).\mathbf{out} \uparrow out(x) \mid \\
\qquad \mathbf{try} \\
\qquad\quad [\![g']\!]_{out_2} \mid \mathbf{in} \downarrow out_2(y).\mathbf{throw}\ x \blacktriangleright [!\ \mathbf{out} \leftarrow val(y)] \\
\qquad \mathbf{catch}\ \mathbf{0})] \\
\quad \mid [\![D]\!])
\end{array}
\tag{A.48.100}
$$

which completes the proof for this case.

(**Case** $[\![p.S(c)]\!]_{out}$)
We consider the possible transitions of

$$(\mathtt{new}\,\widetilde{n})(\mathtt{instance}\,p \blacktriangleright S \Leftarrow (\mathtt{out} \leftarrow args(c).\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)) \mid [\![D]\!]) \quad \text{(A.48.101)}$$

We have that

$$\begin{array}{l}(\mathtt{new}\,\widetilde{n})(\mathtt{instance}\,p \blacktriangleright S \Leftarrow (\mathtt{out} \leftarrow args(c).\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)) \mid [\![D]\!]) \\ \overset{(\nu u)p \blacktriangleright def\ S}{\Rightarrow} \\ (\mathtt{new}\,\widetilde{n})(u \blacktriangleleft [\mathtt{out} \leftarrow args(c).\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)] \mid [\![D]\!])\end{array} \quad \text{(A.48.102)}$$

We have that

$$p.S(c) \overset{p.S<c,u>}{\longrightarrow} ?u \quad \text{(A.48.103)}$$

We derive

$$\begin{array}{l}(\mathtt{new}\,\widetilde{n})(u \blacktriangleleft [\mathtt{out} \leftarrow args(c).\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)] \mid [\![D]\!]) \\ \overset{u \blacktriangleleft \leftarrow args(c)}{\Rightarrow} \overset{\approx}{} \\ (\mathtt{new}\,\widetilde{n})([\![?u]\!]_{out} \mid [\![D]\!])\end{array} \quad \text{(A.48.104)}$$

which completes the proof for this case.
(**Case** $[\![?u]\!]_{out}$)
We consider the possible transitions of

$$(\mathtt{new}\,\widetilde{n})(u \blacktriangleleft [\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)] \mid [\![D]\!]) \quad \text{(A.48.105)}$$

We have

$$\begin{array}{l}(\mathtt{new}\,\widetilde{n})(u \blacktriangleleft [\mathtt{in} \leftarrow result(x).\mathtt{out} \uparrow out(x)] \mid [\![D]\!]) \\ u \blacktriangleleft \leftarrow result(c) \\ \overset{}{\Rightarrow} \\ (\mathtt{new}\,\widetilde{n})(u \blacktriangleleft [\mathtt{out} \uparrow out(c)] \mid [\![D]\!])\end{array} \quad \text{(A.48.106)}$$

We have that

$$?u \xrightarrow{u?c} let(c) \quad \text{(A.48.107)}$$

Considering Proposition 5.4(5) and 5.4(4) we conclude

$$(\mathtt{new}\,\widetilde{n})(u \blacktriangleleft [\mathtt{out} \uparrow out(c)] \mid [\![D]\!]) \approx (\mathtt{new}\,\widetilde{n})([\![let(c)]\!]_{out} \mid [\![D]\!]) \quad \text{(A.48.108)}$$

which completes the proof for this case.
(**Case** $[\![let(c)]\!]_{out}$)
We consider the possible transitions of

$$(\mathtt{new}\,\widetilde{n})(\mathtt{out} \downarrow out(x) \mid [\![D]\!]) \quad \text{(A.48.109)}$$

We have

$$(\mathtt{new}\,\widetilde{n})(\mathtt{out} \downarrow out(x) \mid [\![D]\!]) \overset{\downarrow out(c)}{\Rightarrow} (\mathtt{new}\,\widetilde{n})(\mathtt{stop} \mid [\![D]\!]) \quad \text{(A.48.110)}$$

We have that

$$let(c) \xrightarrow{!c} 0 \quad \text{(A.48.111)}$$

Since $[\![0]\!]_{out} \triangleq \mathtt{stop}$ the proof for this case is complete.
(**Case** $[\![n.S(c)]\!]_{out}$)

We consider the possible transitions of

$$(\mathbf{new}\,\widetilde{n})(\mathbf{instance}\,n \blacktriangleright S \Leftarrow (\mathbf{out} \leftarrow args(c).!\,\mathbf{in} \leftarrow result(x).\mathbf{out} \uparrow out(x)) \mid [\![D]\!]) \quad \text{(A.48.112)}$$

We have that $n.S(x) = e$ is defined in D, which is encoded by

$$[\![n.S(x) = e]\!] \triangleq n \blacktriangleright [!\,\mathbf{def}\,S \Rightarrow (\mathbf{in} \leftarrow args(x).[\![e]\!]_{out} \mid !\,\mathbf{in} \downarrow out(x).\mathbf{out} \leftarrow result(x))] \quad \text{(A.48.113)}$$

Considering (A.48.113) we derive

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})(\mathbf{instance}\,n \blacktriangleright S \Leftarrow (\mathbf{out} \leftarrow args(c).!\,\mathbf{in} \leftarrow result(x).\mathbf{out} \uparrow out(x)) \mid [\![D]\!]) \\ \stackrel{\tau}{\Rightarrow} \\ (\mathbf{new}\,\widetilde{n})((\mathbf{new}\,a)(a \blacktriangleleft [!\,\mathbf{in} \leftarrow result(x).\mathbf{out} \uparrow out(x)] \\ \qquad\qquad \mid a \blacktriangleright [[\![e]\!]_{out}\{x \leftarrow c\} \mid !\,\mathbf{in} \downarrow out(x).\mathbf{out} \leftarrow result(x)]) \mid [\![D]\!])\end{array} \quad \text{(A.48.114)}$$

We have that

$$n.S(c) \xrightarrow{\tau} e\{x \leftarrow c\} \quad \text{(A.48.115)}$$

From Lemma A.44 we conclude

$$\begin{array}{l}(\mathbf{new}\,\widetilde{n})((\mathbf{new}\,a)(a \blacktriangleleft [!\,\mathbf{in} \leftarrow result(x).\mathbf{out} \uparrow out(x)] \\ \qquad\qquad \mid a \blacktriangleright [[\![e]\!]_{out}\{x \leftarrow c\} \mid !\,\mathbf{in} \downarrow out(x).\mathbf{out} \leftarrow result(x)]) \mid [\![D]\!]) \\ \approx \\ (\mathbf{new}\,\widetilde{n})([\![e]\!]_{out}\{x \leftarrow c\} \mid [\![D]\!])\end{array} \quad \text{(A.48.116)}$$

which completes the proof for this last case.

Proof of Proposition 4.1

Proof. We start by proving the $\Longrightarrow$ direction. We have

$$O \xrightarrow{l_1} O_1 \xrightarrow{l_2} O_2 \xrightarrow{l_3} \ldots \xrightarrow{l_k} O_k \xrightarrow{l_{k+1}} \ldots \quad \text{(A.48.117)}$$

From (A.48.117) and Lemma A.47 we conclude

$$P_0 \stackrel{match_{out}(l_1)}{\Rightarrow} P_1 \quad \text{(A.48.118)}$$

and

$$P_1 \approx (\mathbf{new}\,\widetilde{n})([\![O_1]\!]_{out} \mid [\![D]\!]) \quad \text{(A.48.119)}$$

Again from (A.48.117) and Lemma A.47 we conclude

$$([\![O_1]\!]_{out} \mid [\![D]\!]) \stackrel{match_{out}(l_2)}{\Rightarrow} P_2' \quad \text{(A.48.120)}$$

and

$$P_2' \approx (\mathbf{new}\,\widetilde{n})([\![O_2]\!]_{out} \mid [\![D]\!]) \quad \text{(A.48.121)}$$

From (A.48.119) and (A.48.120) we conclude

$$P_1 \stackrel{match_{out}(l_2)}{\Rightarrow} P_2 \quad \text{(A.48.122)}$$

and

$$P_2 \approx P_2' \quad \text{(A.48.123)}$$

From (A.48.121) and (A.48.123) we have

$$P_2 \approx (\mathtt{new}\, \widetilde{n})([\![O_2]\!]_{out} \mid [\![D]\!]) \tag{A.48.124}$$

We have that (A.48.122) and (A.48.124) are obtained from (A.48.118) and (A.48.119), and can be used as premises as (A.48.118) and (A.48.119) were. Following this reasoning we can proceed indefinitely and obtain

$$P_0 \overset{match_{out}(l_1)}{\Rightarrow} P_1 \overset{match_{out}(l_2)}{\Rightarrow} P_2 \overset{match_{out}(l_3)}{\Rightarrow} \ldots \overset{match_{out}(l_k)}{\Rightarrow} P_k \overset{match_{out}(l_{k+1})}{\Rightarrow} \ldots \tag{A.48.125}$$

Proof for the $\Longleftarrow$ direction follows analogously from Lemma A.48.